Evaluation of Subgrid-Scale Turbulence Models Using a Fully Simulated Turbulent Flow

BY

R. A. Clark,
J. H. Ferziger,
AND
W. C. Reynolds

Prepared from work done under Grant

NASA-NgR-05-020-622

Report No. TF-9

Thermosciences Division
Department of Mechanical Engineering
Stanford University
Stanford, California

March 1977

Acknowledgments

The author gratefully acknowledges the helpful comments of Drs. J. Ferziger, W. Reynolds, A. Leonard, and J. Sicilian. I appreciate the speedy typing of the final copy by Ruth Korb. I am especially thankful for the patient encouragement from my wife Catherine in the face of many unavoidable delays in the completion of this work.

This work was partially supported by NASA-Ames Research Center under Grant NgR-020-05-622.

Table of Contents

List of Tables

List of Figures

EVALUATION OF SUBGRID-SCALE TURBULENCE MODELS
USING A FULLY SIMULATED TURBULENT FLOW

Abstract

Numerous models have been proposed for approximating the subgrid-scale Reynolds stresses in numerical simulations of turbulent fluid flow. Until now, the only way to verify such approximations has been to observe the resulting behavior of the large-scale flow. If the entire turbulent flow field were known, it would be possible to make direct comparisons between the exact Reynolds stresses and a given model. We have calculated an "exact" turbulent flow field on a three-dimensional grid with 64 points on a side. The flow simulates grid-generated turbulence from wind tunnel experiments. In this simulation, the grid spacing is small enough to include essentially all of the viscous energy dissipation and the box is large enough to contain the largest eddy in the flow. The method is limited to low-turbulence Reynolds numbers, in our case $R_\lambda = 36.6$.

In order to complete the calculation using a reasonable amount of computer time with reasonable accuracy, we developed a third-order time-integration scheme which runs at about the same speed as a simple first-order scheme. It obtains this accuracy by saving the velocity field and its first-time derivative at each time step. Fourth-order accurate space-differencing is used.

The results of this simulation were treated as an experimental realization of physical turbulence. We then superimposed an $8 \times 8 \times 8$ coarse mesh over the original fine mesh and defined a filtered velocity field $\bar{u}_i(\underline{x})$ as the local spatial average of u_i. From these we defined the subgrid-scale velocity field u_i by $u_i = \bar{u}_i + u_i'$. The filtering process gives rise to three terms in the Navier-Stokes equations. These are the Reynolds stress, $\overline{u_i'u_j'}$, the Leonard term, $\overline{\bar{u}_i\bar{u}_j} - \bar{u}_i\bar{u}_j$, and the cross term, $\overline{u_i'\bar{u}_j} + \overline{\bar{u}_i u_j'}$. We demonstrated that the cross term is non-zero and is, in fact, dissipative; we also developed a model for it. The Leonard term and the cross term can be combined into a single term which can be modeled by $(\vec{\nabla}\bar{u}_i)\cdot(\vec{\nabla}\bar{u}_j)$. This reduces, in one dimension, to a quadratic artificial viscosity frequently used in compressible flow calculations

The relationship between the filtering process and artificial viscosity is shown.

Finally we calculated each of the above terms within each cell on the coarse mesh, and we attempted to model them using the filtered velocity field. For each model we calculated the correlation between the model and its "exact" value. We found the correlation between the Leonard and cross terms and their models to be excellent, around ninety percent. The correlation between model and experiment for the Reynolds stress is not as good, but we did achieve about seventy percent correlation between the dissipation produced by the Reynolds stress and its model. We found no model that is significantly better than the standard Smagorinsky model. We found that models using the subgrid-scale turbulent kinetic energy are no better than Smagorinsky's, even when we had the exact subgrid-scale kinetic energy to work with. All of these conclusions must be qualified by stating that we were working at very low turbulent Reynolds numbers, and the results cannot necessarily be extrapolated to higher Reynolds numbers.

Chapter I

INTRODUCTION

We begin with a brief discussion of the general approach to the numerical simulation of turbulent flows. It is generally not possible to calculate a turbulent flow in complete detail, because the range of length scales involved is so large that the amount of data that would have to be handled is orders of magnitude greater than the capacity of any existing or projected computer. For this reason, the traditional approaches to such problems have been based on Reynolds' original idea of averaging the Navier-Stokes equations over an ensemble of identical flows or an appropriate interval of time or space. One then has equations for an averaged velocity field $\overline{u}(x,t)$, where the overbar denotes averaging according to whatever definition is employed. If we then define a fluctuating velocity component by $u(x,t) = \overline{u}(x,t) + u'(x,t)$, the averaged equations can be written (for an incompressible flow with constant viscosity)

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial}{\partial x_j} \overline{u}_i \overline{u}_j = - \frac{\partial \overline{p}}{\partial x_i} + \nu \nabla^2 \overline{u}_i - \frac{\partial}{\partial x_j} R_{ij} \quad , \tag{1.1}$$

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0 \quad , \tag{1.2}$$

$$R_{ij} = \overline{u_i' u_j'} \quad , \tag{1.3}$$

since $\overline{\overline{u}_i} = \overline{u}_i$ and $\overline{u_i'} = 0$ as a consequence of the definition of averaging. To formally close this system of equations it is necessary to find an expression for R_{ij} (the Reynolds stress) in terms of $\overline{u}_i$.

The search for such expressions (closure models) has been a major direction in turbulence work for many decades. Prior to the advent of computers, only very simple models could be used, i.e., those which yielded equations to which one could obtain solutions analytically. These models, which assume that the Reynolds stresses, like viscous stresses, are proportional to the local strain rate of the fluid, give acceptable results

provided the range of flows they are required to predict is not too large These models (eddy viscosity models) can be adjusted for particular flows to produce excellent results. The problem is that this type of model must be adjusted to experimental data and can therefore be used only in an interpolative manner

More advanced models have been proposed since the introduction of large digital computers. These include models in which the eddy viscosity is a function of space and/or time (turbulence kinetic energy and two-equation models) and still more advanced models which utilize partial differential equations for the Reynolds stresses are currently being developed. It is too early to predict the success of these approaches, but there is reason to believe that their range of application will be limited. By this we mean that the model parameters will probably need to be adjusted for each type of flow.

To see why this might be the case and what might be done about it, consider the overall nature of a turbulent flow field. The range in length scales between the largest and smallest turbulent structures is many orders of magnitude in most flows of interest The largest turbulent structures draw energy from the mean flow. This energy is thought to cascade through an intermediate range of eddy sizes to eventually reach the smallest turbulent eddies. The smallest eddies then dissipate the kinetic energy to internal energy by viscous effects. There is reason to believe that the largest structures in a turbulent flow are much more dependent on the origin of the turbulence, i.e., the type of flow under consideration, than either the intermediate or small-scale structures. This could explain the failure of any single model to predict a wide variety of flows when the large-scale turbulence is included in $\overline{u_i' u_j'}$. It appears more likely that a universal model might be found if only the intermediate and small-scale turbulence is modeled. This approach (large eddy simulation) requires that the large-scale turbulence be calculated explicitly and that the small scales (the "subgrid" scale turbulence) be modeled. With the recent advances in high-speed computing machinery, this approach has become increasingly practical in the last several years (Hirt, 1969; Deardorff, 1970; Fox and Lilly, 1972).

In large eddy simulation, one averages the Navier-Stokes equations over a small spatial region in order to remove the small-scale fluctuations. The resulting equation for the large-scale field contains a term similar to, but more complicated than, the Reynolds stress R_{ij} of Eqs. (1 1) and (1.3), and this term (the subgrid scale Reynolds stress) must be modeled.

Several models for the subgrid scale R_{ij} have been proposed. The problem has been how to verify a proposed model. The best that could be done until now was to compare the evolution of the large-scale structures in a computation to those in an experiment. This will not reveal whether or not the actual subgrid scale Reynolds stress is being accurately modeled, but only whether or not the subgrid scale Reynolds stress and the model have the same net effect on the large-scale motions for the particular type of flow in question. In addition, virtually all models contain at least one adjustable constant which must be set by some ad hoc assumption or by adjusting the constant to fit some important aspect of an experiment. On the other hand, if there were a physical experiment which measured everything of interest in a turbulent flow field, from the largest turbulent structure to the smallest eddy, it would then be possible to compute the subgrid scale R_{ij} exactly, and then compare its value at each point in space to the prediction of a model. Unfortunately, there is no laboratory experiment capable of such measurements. But if we could compute the evolution of a turbulent flow field on a sufficiently fine grid (fine enough to include all of the turbulent structures) numerically, then we would have all the information necessary to make direct comparisons between measured values of the subgrid R_{ij} and the model predictions.

The objectives of this work were to accurately calculate a three-dimensional turbulent flow field on a fine grid by directly integrating the Navier-Stokes equations using no approximations with respect to the structure of the turbulence, i.e., without having to average the equations, and then to use the results of that calculation to examine subgrid scale models on a coarse mesh overlayed on the original fine mesh. Practical limitations require that this be done at a low Reynolds number, since at high Reynolds numbers the difference in scale between the largest and smallest eddies is so great that computer simulation is impractical. Also, the flow should be as physically simple as possible (see below)

In Chapter II we describe the numerical integration method which was used to compute the flow field on the fine mesh. A third-order time scheme is introduced which has not previously been used in this application Because this is the first attempt at model verification, we have chosen the simplest possible turbulent flow field for our calculation, homogeneous isotropic turbulence This avoids any problems of anisotropy, but restricts the results to problems in which the subgrid scale turbulence can be assumed isotropic In Chapter III we describe in detail how the main calculation was performed, including some programming techniques we employed to greatly speed up the calculation In Chapter IV we discuss the general problem of modeling subgrid scale turbulence with emphasis on methods which could be verified by our calculations. In Chapter V we demonstrate that the results of our main calculation do in fact have the properties of real turbulent flow. We then make the comparisons of the numerically calculated values of R_{ij} and the predictions of various models and show that, although the models currently used are not as accurate as one would like, they are difficult to improve upon in a simple manner.

Chapter II

THE NUMERICAL METHOD

2.1 General

The finite-difference scheme chosen for the main calculation is fourth-order accurate in space and third-order accurate in time. The rationale for this choice is worth a brief discussion.

In many large computer simulations a major effort is made to keep the problem small enough to be contained entirely in the main memory of the computer. This is done to avoid the use of the disk memory and its relatively slow rate of data transfer. The problem we shall try to solve has three velocity components at each of 262,144 grid points, for a total of 786,432 words necessary to specify the velocity field at one time step. This number alone exceeds the roughly 400,000 words of memory available in our CDC 7600 large-core memory. Since we are forced to utilize disk memory, waiting times for the completion of data transfer to and from disk become a major consideration. Large amounts of data must be transferred from disk to main memory, processed in main memory, then transferred back to disk. If the processing time is too short, the data transfer time will determine the running time of the problem. In our case, using fourth-order differencing in space, we found that only about five percent of the total running time was spent in waiting for data transfer to be completed. This means that the data were processed slightly faster than it could be transferred, even though a double-buffering scheme was used Had we used second-order space differencing, we would not have gained anything in running time, since the data-transfer time would still be the same Reducing the processing time would have simply increased the percentage of wait time. This means that we have used fourth-order space differencing with no increase in computer charges with respect to second-order differencing, i.e., increased accuracy is obtained at essentially no cost. We emphasize that this choice is not made for accuracy reasons, but is a result of being forced to use the disk memory.

Having settled on fourth-order accuracy in space, we must next decide how to handle the time differencing. In any numerical method, common sense dictates that the truncation error due to the time differencing should be roughly the same as the truncation error due to the space differencing This can be done even with first-order time differencing if the time step used is sufficiently small. The criterion for choosing a time-differencing scheme now becomes cost, i.e., computer running time. If a second-order time scheme were to take twice the computing time per time step as a first-order scheme but allowed us to increase the time step by more than a factor of two, the total running time would be reduced and the second-order scheme would be justified.

You generally get what you pay for with numerical methods. If greater accuracy is desired, you must pay for it. However, there can sometimes be more than one method of payment The most common method of payment is in increased running time. For example, the simplest two-step, second-order schemes (Roache, 1972) obtain second-accuracy by essentially performing a simple first-order scheme twice, thus doubling the running time. Another method of payment can be in increased storage requirements. The leap-frog scheme performs essentially the same calculations as a first-order scheme, but it obtains second-order accuracy by saving an extra time step in the calculation, hence doubling the storage requirements. On most present-day computing systems, the extra charge for doubling the storage requirement is small in comparison to the savings resulting from halving the CPU time. On this basis the second-order accuracy of the leap-frog method appears to be obtained almost for free. Another way to look at it is that the user is usually charged for most of the available storage, and if he does not use it he is short-changing himself.

With the above in mind, we note that since our problem requires use of the disk the storage available is, for all practical purposes, unlimited. As noted above, the leap-frog scheme obtains its second-order accuracy by saving the velocity field at an extra time step. We have developed a time-differencing scheme that obtains third-order accuracy by saving the first time derivative of the velocity field, as well as the velocity field itself. With this method the running time per time step is essentially unchanged from first-order methods, or from the leap-frog method, but the

time step can be increased while maintaining the same truncation error. This reduces the total running time of the problem without reducing the accuracy.

2.2 The Time-Differencing Scheme

We now develop the third-order method that we will use.* It is essentially a predictor-corrector method with a second-order leap-frog predictor and a third-order Adams-Moulton corrector. To explain the method we deal with the ordinary differential equation

$$u_t = \alpha u \tag{2.1}$$

which has the exact solution $u = \exp(\alpha t)$. Suppose that we are given u at times $n\delta t$ and $(n-1)\delta t$ to third-order accuracy, i.e.,

$$u^n = \exp[\alpha n\delta t] + O(\delta t^4) \tag{2.2a}$$

$$u^{n-1} = \exp[\alpha(n-1)\delta t] + O(\delta t^4) \tag{2.2b}$$

and $(u^*)_t$ at time $n\delta t$ and $(n-1)\delta t$ to second-order accuracy, i.e.,

$$(u^*)_t^n = \alpha \exp[\alpha n\delta t] + O(\delta t^3) \tag{2.3a}$$

$$(u^*)_t^{n-1} = \alpha \exp[\alpha(n-1)\delta t] + O(\delta t^3) \tag{2.3b}$$

Thus, $(u^*)^n$ and u^n represent two numerical approximations to the exact solution; they are, respectively, the predicted and corrected values. Let us first approximate the solution to equation (2.1) at time $(n+1)\delta t$ using the standard leap-frog method.

$$(u^*)^{n+1} = u^{n-1} + 2\delta t(u^*)_t^n \tag{2.4}$$

Using a Taylor series expansion to get $\exp(\alpha(n+1)\delta t)$ in terms of $\exp(\alpha n\delta t)$ and introducing the notation $\gamma = \alpha\delta t$, we have

* For a general discussion of methods of this type, see Appendix A

$$\exp[\alpha(n\pm1)\delta t] = \left(1 \pm \gamma + \frac{\gamma^2}{2} \pm \frac{\gamma^3}{6} + \frac{\gamma^4}{24} \pm \ldots\right) \exp[\alpha n\delta t] \qquad (2.5)$$

from which it is easy to show that

$$(u^*)^{n+1} = \exp[\alpha(n+1)\delta t] - \frac{\gamma^3}{3} \exp[\alpha n\delta t] + O(\delta t^4) \qquad (2.6)$$

Hence, $(u^*)^{n+1}$ and $(u^*)_t^{n+1} = \alpha(u^*)^{n+1}$ are accurate to second-order. Now that we have $(u^*)_t^{n+1}$, $(u^*)_t^n$, and $(u^*)_t^{n-1}$ to second-order, we can evaluate $(u^*)_{tt}^n$ to first-order and $(u^*)_{ttt}^n$ to zeroth order.

$$(u^*)_{tt}^n = \frac{(u^*)_t^{n+1} - (u^*)_t^{n-1}}{2\delta t} = \alpha^2 \exp(\alpha n\delta t) + O(\delta t^2) \qquad (2.7)$$

$$(u^*)_{ttt}^n = \frac{(u^*)_t^{n+1} - 2(u^*)_t^n + (u^*)_t^{n-1}}{\delta t^2} = \alpha^3 \exp(\alpha n\delta t) + O(\delta t) \qquad (2.8)$$

Next we evaluate the corrected value u^{n+1} to third-order accuracy by using the expansion (2.5) in the form

$$u^{n+1} = u^n + \delta t(u^*)_t^n + \frac{\delta t^2}{2}(u^*)_{tt}^n + \frac{\delta t^3}{6}(u^*)_{ttt}^n \qquad (2.9)$$

which is identical to

$$u^{n+1} = [\exp(\alpha n\delta t) + O(\delta t^4)] + \delta t[\alpha \exp(\alpha n\delta t) + O(\delta t^3)] \qquad (2.10)$$
$$+ \frac{\delta t^2}{2}[\alpha^2 \exp(\alpha n\delta t) + O(\delta t^2)] + \frac{\delta t^3}{6}[\alpha^3 \exp(\alpha n\delta t) + O(\delta t)]$$

Comparing (2.5) and (2.10), we see that

$$u^{n+1} = \exp[\alpha(n+1)\delta t] + O(\delta t^4)$$

Summarizing the method, the predictor step is leap frog

$$(u^*)^{n+1} = u^{n-1} + 2\delta t(u^*)_t^n \qquad (2\ 11)$$

and the corrector step in simplified form is

$$u^{n+1} = u^n + \frac{2}{3}\delta t(u^*)_t^n + \frac{5}{12}\delta t(u^*)_t^{n+1} - \frac{1}{12}(u^*)_t^{n-1} \qquad (2\ 12)$$

which will be recognized as an implicit Adams-Moulton method. It is, in fact, possible to produce a fourth-order method in this way, but the extra advantage is minimal.

2.3 Numerical Stability of the Third-Order Method

Following standard Von Neumann analysis (Richtmyer, 1967), we seek solutions to equations (2.11) and (2.12) of the form

$$u^n = A^n u \qquad (2.13a)$$

$$(u^*)^n = A^n u^* \qquad (2.13b)$$

where A is in general a complex constant. The numerical method will be stable if we can ensure that $|A| \leq 1$ for $Re(\alpha) \leq 0$. First, we replace $(u^*)_t^n$ by $\alpha(u^*)^n$ in Eq. (2.11) to get

$$u^{n-1} = (u^*)^{n+1} - 2\gamma(u^*)^n \qquad (2.14)$$

where again $\gamma = \alpha\delta t$. Changing the index, we have

$$u^n = (u^*)^{n+2} - 2\gamma(u^*)^{n+1} \qquad (2\ 15a)$$

and

$$u^{n+1} = (u^*)^{n+3} - 2\gamma(u^*)^{n+2} \qquad (2\ 15b)$$

Substituting (2.15a) and (2.15b) into (2 12) and again replacing $(u^*)_t^n$ by $\alpha(u^*)^n$ yields

$$(u^*)^{n+3} - 2\gamma(u^*)^{n+2} = (u^*)^{n+2} - 2\gamma(u^*)^{n+1} + \frac{2}{3}\gamma(u^*)^n + \frac{5}{12}\gamma(u^*)^{n+1} - \frac{1}{2}\gamma(u^*)^{n-1} \qquad (2\ 16)$$

Substituting (2.13b) into (2.16) and multiplying by A^{-n+1} yields a quartic equation for A:

$$A^4 = (1+2\gamma)A^3 + \frac{19}{12}\gamma A^2 - \frac{2}{3}\gamma A + \frac{\gamma}{12} = 0 \qquad (2.17)$$

The stability of the numerical solution is now determined by finding the maximum value of $|\gamma| = |\alpha|\delta t$ with $Re(\alpha) < 0$ for which all four roots of Eq. (2.17) have magnitudes less than 1. This ensures that the solutions given by Eq. (2 13) will not increase exponentially with increasing n in cases in which they should not. The resulting region of stability is shown in Fig. 2.1. The curve shown in Fig. 2.1 is the curve of neutral stability on which the magnitude of the largest A is 1. This curve was found by computing the roots of Eq. (2.17) for fixed γ_i noting the value of $|\gamma_r|$ for which one of the roots becomes unity. Other methods are available, but the simplicity of this calculation does not warrant their use.

2.4 Accuracy of the Third-Order Method

We again consider the ordinary differential equation

$$u_t = \alpha u \qquad (2.18)$$

which has the exact solution

$$u = \exp(\alpha t) = \exp(\gamma n) \qquad (2.19)$$

One of the roots of Eq. (2.17) will be an accurate approximation to $\exp(\gamma)$. The others are parasitic or computational roots. It turns out that the three parasitic roots are highly damped (γ_r-negative and large in comparison to the non-parasitic root), so that the solution obtained will be

$$u(n\delta t) = A_1^n \qquad (2.20)$$

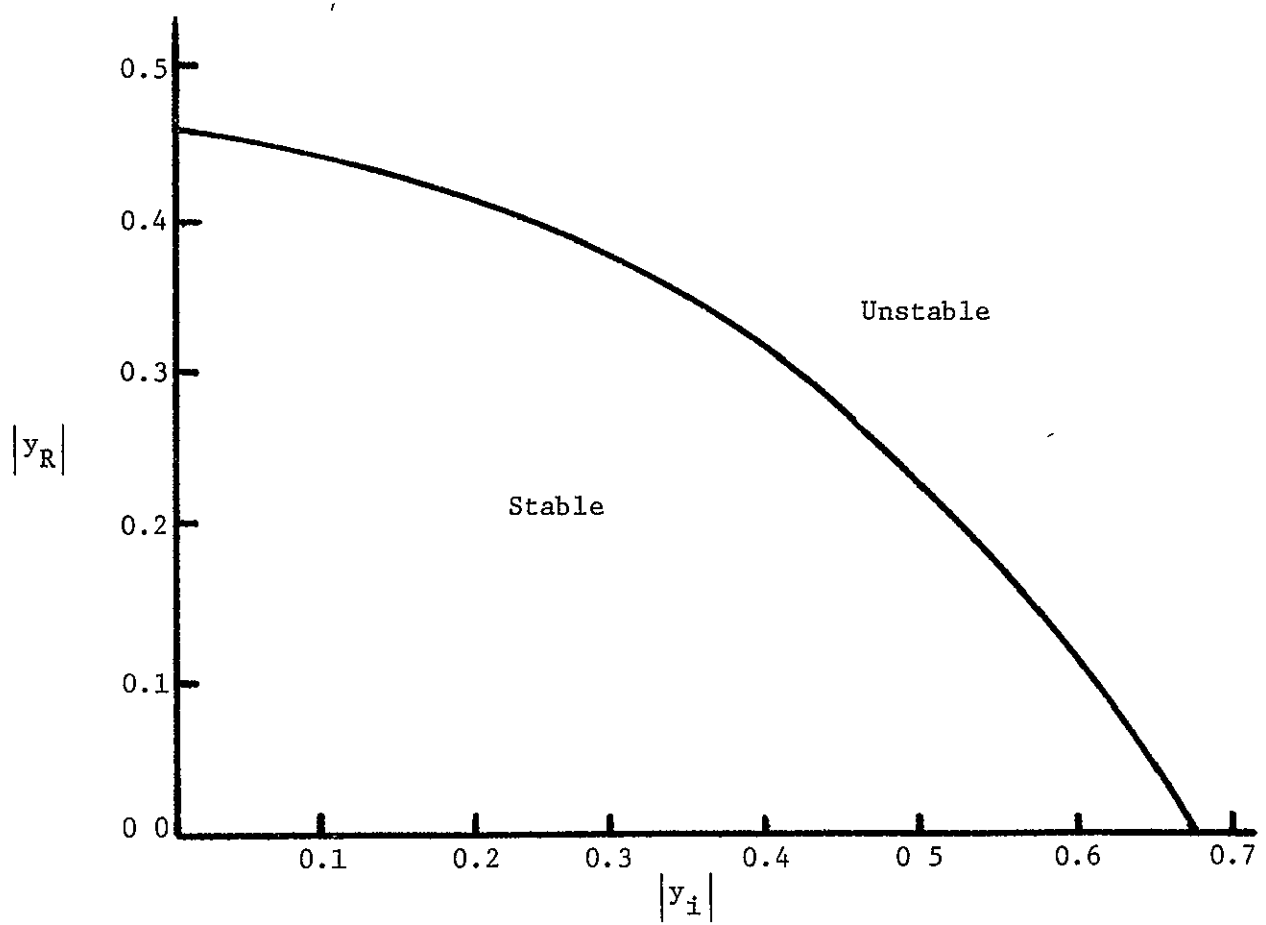


Fig. 2.1 Stability of third-order method

where A_1 is the desired root of Eq (2.17). Let $\gamma_r = Re(\gamma)$ and $\gamma_i = Im(\gamma)$. The accuracy of the solution is determined by how well A_1 approximates $\exp(\gamma)$. We separate the numerical error into the phase and amplitude errors. The phase error is given by

$$\Delta\gamma_p = \{\gamma_i - Im[\ell n(A_1)]\} \tag{2.21}$$

and the amplitude error is given by

$$\Delta\gamma_t = \{\gamma_r - Re[\ell n(A_2)]\} \tag{2.22}$$

We have computed A_1 for the leap-frog and third-order schemes. In Table 2.1 we list some values of γ_i and the error in the computed value of γ_i, $\Delta\gamma_p$, for the leap-frog and third-order schemes with $\gamma_r = 0$. In Table 2.2 we list some values of γ_r and the error in the computed value of γ_i, $\Delta\gamma_r$, for the two schemes with $\gamma_i = 0$. The range of γ_i and γ_r listed in the tables covers the range of interest in our main calculation.

A simple linear equation which is sometimes used as a model for testing numerical approximations to the Navier-Stokes equations is the convected diffusion equation:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

Assuming a solution to this equation of the form $u(x,t) = u(t)e^{ikx}$, we have

$$u_t = (-ick - \nu k^2)u \tag{2.23}$$

The values for mean velocity, viscosity, spatial increment, and time step which correspond to the main calculation described in Chapter III are $c = 5.5$ cm/sec, $\nu = 0.14$ cm^2/sec, $\delta x = 20/64$ cm, $\delta t = 0.0073$ sec, and $0 < k < \pi/\delta x$ Using these values, we have calculated $\gamma_i = -ck\delta t$ and $\gamma_r = -\nu k^2\delta t$ for use in Tables 2.1 and 2.2.

The $\Delta\gamma$'s listed in the tables are the errors per time step. Take, for example, $\gamma_i = 0.201$ (this is equivalent to a wave with wavelength

Table 2.1

Phase Error Comparison

K	$-\gamma_1$	2nd Order $\Delta\gamma_1$	3rd Order $\Delta\gamma_1$
.503	020	$1\ 4 \times 10^{-6}$	7.6×10^{-9}
1.005	.040	1.1×10^{-5}	$2\ 5 \times 10^{-8}$
1.508	040	3.7×10^{-5}	1.9×10^{-7}
2.011	.080	8.8×10^{-5}	8.0×10^{-7}
2 513	.100	1.7×10^{-4}	2.4×10^{-6}
3.016	.120	3.0×10^{-4}	6.0×10^{-6}
3.519	.141	$4\ 8 \times 10^{-4}$	1.2×10^{-5}
4 021	.161	7.2×10^{-4}	$2\ 5 \times 10^{-5}$
4.524	.181	1.0×10^{-3}	4.4×10^{-5}
5.027	.201	1.4×10^{-3}	7.6×10^{-5}
5.529	.221	1.8×10^{-3}	1.2×10^{-4}
6.032	.241	$2\ 4 \times 10^{-3}$	1.8×10^{-4}
6.535	261	3.1×10^{-3}	$2\ 7 \times 10^{-4}$
7.037	.281	3.8×10^{-3}	3.8×10^{-4}
7 540	.302	4.8×10^{-3}	5.2×10^{-4}
8.042	.322	6.0×10^{-3}	7.6×10^{-4}
8.545	.342	7.6×10^{-3}	1.0×10^{-3}
9 048	.362	$8\ 4 \times 10^{-3}$	1.3×10^{-3}
9.550	380	$1\ 0 \times 10^{-2}$	1.7×10^{-3}
10.053	402	1.2×10^{-2}	2.1×10^{-3}

Table 2.2

Amplitude Error Comparison

K^2	$-\gamma_r$	2nd Order $\delta\gamma_r$	3rd Order $\delta\gamma_r$
.252	2.82×10^{-4}	3.8×10^{-12}	3.8×10^{-15}
1.011	1.13×10^{-3}	2.5×10^{-10}	4.8×10^{-13}
2.274	$2\ 55 \times 10^{-3}$	2.8×10^{-9}	1.2×10^{-11}
4.043	4.53×10^{-3}	1.6×10^{-8}	1.2×10^{-10}
6.312	7.07×10^{-3}	5.9×10^{-8}	7.4×10^{-10}
9.096	1.02×10^{-2}	1.8×10^{-7}	3.1×10^{-9}
12.380	1.39×10^{-2}	4.5×10^{-7}	1.1×10^{-8}
16.170	1.81×10^{-2}	9.9×10^{-7}	3.1×10^{-8}
20.465	2.29×10^{-2}	2.0×10^{-6}	8.2×10^{-8}
25.266	2.83×10^{-2}	3.8×10^{-6}	$1\ 9 \times 10^{-7}$
30.572	3.42×10^{-2}	6.7×10^{-6}	4.1×10^{-7}
36.383	4.07×10^{-2}	1.1×10^{-5}	6.3×10^{-7}
42.700	4.78×10^{-2}	1.8×10^{-5}	1.6×10^{-6}
49.522	5.55×10^{-2}	2.8×10^{-5}	2.8×10^{-6}
56.849	6.37×10^{-2}	$4\ 3 \times 10^{-5}$	5.0×10^{-6}
64.681	7.24×10^{-2}	6.3×10^{-5}	8.5×10^{-6}
73.020	$8\ 07 \times 10^{-2}$	$9\ 1 \times 10^{-5}$	1.3×10^{-5}
81.862	9.17×10^{-2}	1.2×10^{-4}	2.1×10^{-5}
91.211	$1\ 02 \times 10^{-1}$	1.8×10^{-4}	$3\ 5 \times 10^{-5}$
101.065	1.13×10^{-1}	2.4×10^{-4}	5.3×10^{-5}

$4\delta x$ in our main calculation). For $\gamma_1 = 0.201$, the phase error per time step is 18 times larger for the leap-frog scheme than for the third-order scheme. This means that if both schemes used the same time step the accumulated error at a given point in time would be 18 times greater using leap frog. This is not the whole story The relevant question to ask is how much would you have to reduce the time step using the leap-frog scheme in order to get the same error as with the third-order scheme? Suppose we are using the third-order scheme with a δt such that $\gamma_1 = 0.201$; this value is typical of the problem that we actually ran. If we reduce the time step by a factor of 4.4 and use the leap-frog scheme, the phase error per time step will be 1.9×10^{-5}, but we will have to run 4.4 times as many time steps so that the phase error per original time step will be $4.4 \times 1.9 \times 10^{-5} = 8.36 \times 10^{-5}$, which is slightly larger than the 7.6×10^{-5} for the third-order scheme. For smaller γ_1 the factors are larger than 4.4, and for larger γ_1 the factors are smaller than 4.4 Our conclusion is that for the leap-frog scheme to achieve the same accuracy as the third-order scheme the time step would have to be reduced by at least a factor of four, thus increasing the running time of the problem by nearly a factor of four.

To further demonstrate the accuracy of the third-order scheme, we present, in Fig. 2.2, the results of a numerical test on the one-dimensional wave equation:

$$u_t + cu_x = 0 \tag{2.24}$$

u_x was calculated by the use of Fourier transforms so that the only error in the numerical solution is due to the time-differencing scheme. u was defined at 64 evenly spaced points and was initially zero, except for the triangle shown near the center. Periodic boundary conditions were applied, and we set $u\delta t/\delta x = 0.2$. In the exact solution, the triangle moves from left to right at the constant speed c so that after 1600 time steps the triangle should have swept across the grid five times and the exact solution is identical to the initial conditions The third-order calculation used 3% more computing time than the leap-frog calculation and twice the amount of storage. The starting conditions in each case were exact This

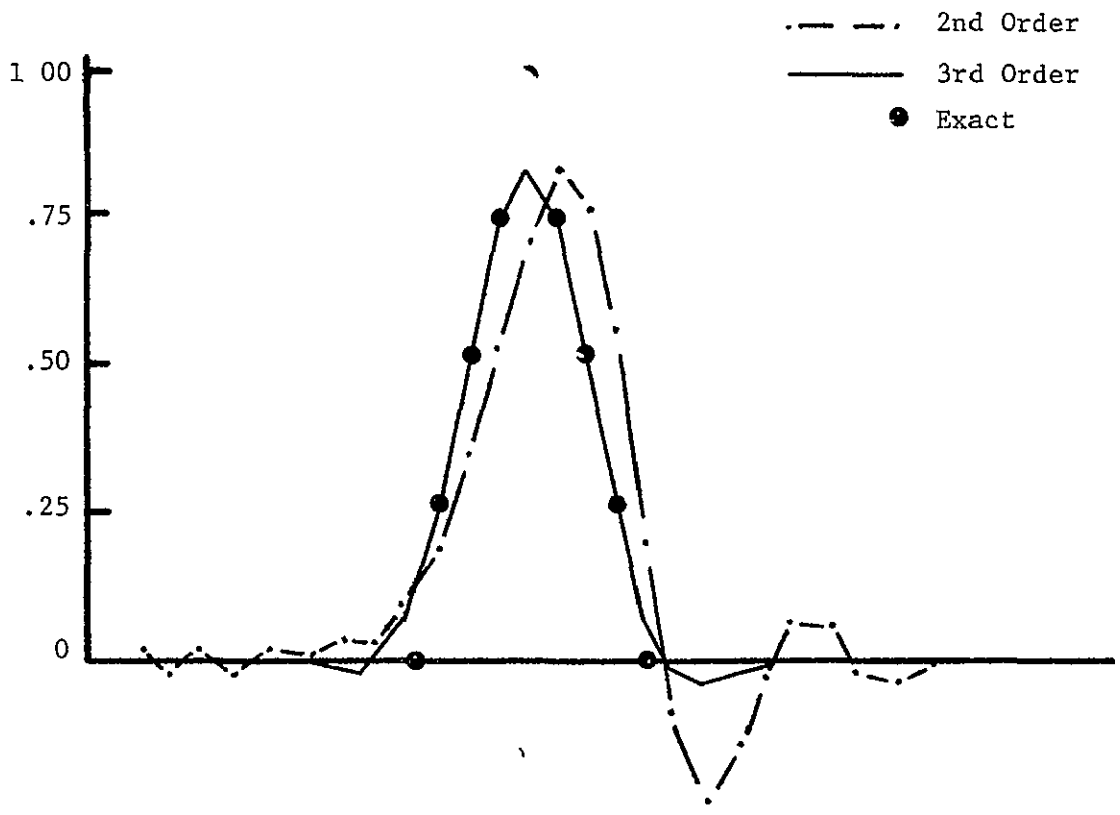


Fig 2 2 Solutions to $u_t + cu_x = 0$.

was done by calculating the Fourier transform of the initial conditions, multiplying each Fourier component by the appropriate $A_1(ick\delta t)$ and inverting the resulting Fourier field to get the value of u at time δt.

2.5 Starting the Third-Order Method

Given the initial conditions for a problem at time $t = 0$, we cannot get the solution at time $t = \delta t$ by the third-order method. The field at $t = \delta t$ must be found by another method. The technique we used is to divide the initial time step δt into three increments of $\delta t/3$ and use a two-step predictor corrector method to obtain $u(\delta t)$ to second-order accuracy. We tested this method by again solving the equation $u_t + cu_x = 0$ as above, except that rather than using the exact numerical solution for $t = \delta t$ we used the second-order predictor corrector scheme to get the second time step. After continuing for 1600 time steps, the two solutions agreed at every point to three significant figures.

Although no difficulties are encountered in starting the scheme in the linear case, we found a weak nonlinear instability while solving the Navier-Stokes equations. Its onset could be detected by the values of u^n and $(u^*)^n$ slowly diverging. The problem was cured by setting $(u^*)^n_t = u^n_t$ at the end of the first few time steps. The calculation of u^n_t (which is not needed elsewhere) almost doubles the computing time for each time step where it is needed In a test calculation on a $16 \times 16 \times 16$ mesh, we found it was sufficient to make this correction after the first four time steps after which no further evidence of instability was seen for the next 130 time steps. In the main $64 \times 64 \times 64$ calculation, we alternately turned the correction on for four time steps and off for four time steps throughout the problem. An examination of the skewness of the velocity field as a function of time (Section 5.1) leads us to believe that the correction could have been turned off permanently after eight applications.

2.6 Space Differencing

The time-integration scheme has been presented in terms of the ordinary differential equation

It is equally applicable to the system of incompressible Navier-Stokes equations

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x_j}(u_i u_j) = -\frac{\partial p}{\partial x_i} + \nu \nabla^2 u_i \tag{2.25a}$$

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2.25b}$$

if the spatial derivatives are calculated at least as accurately as the time derivatives. The method for calculating $\partial u_i/\partial t$ and maintaining zero numerical divergence will be discussed in Chapter III. The spatial derivatives were approximated by fourth-order accurate spatial differencing. This means that the equations we are actually solving are

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x_j}(u_i u_j) = -\frac{\partial p}{\partial x_i} + \nu \nabla^2 u_i + O(\delta t^3) + O(\delta x^4) \tag{2.26a}$$

$$\frac{\partial u_i}{\partial x_i} = O(\delta t^3) + O(\delta x^4) \tag{2.26b}$$

To properly compare error terms, we must include the velocity c to be dimensionally consistent. When we compare terms of $O(\delta t^n)$ to terms of $O(\delta x^n)$, we really need to look at terms like $(c\delta t)^n$ and δx^n. The choice of the appropriate value of c to use in the nonlinear case is somewhat unclear in a turbulent flow simulation, but the r.m.s. velocity is probably a reasonable guess. However, the stability condition gives essentially an upper limit on $c\delta t/\delta x$, the Courant number, where now the maximum value of u must be used for c to assure safety. The result is that δt is normally chosen so that the time-differencing error is in fact somewhat smaller than the space-differencing error.

Now we consider conservation of momentum and energy. For simplicity we have chosen to use centered, fourth-order spatial differencing in space with all quantities cell-centered. The scheme which was used in the main calculation was

$$(u_i)_t = -\frac{1}{2}(D_j u_i u_j + u_j D_j u_i) - D_i p + \nu D_k^2 u_i \tag{2.27}$$

where

$$D_j = \text{a fourth-order approximation to } \partial/\partial x_j,$$

$$D_j u_i = \frac{u_i(j-2) + 8[u_i(j+1) - u_i(j-1)] - u_i(j+2)}{12\delta x_j} \tag{2.28}$$

and

$$D_k^2 = \text{a fourth-order approximation to } \nabla^2:$$

$$\begin{aligned} D_k^2 u_i = \{&16[u_i(i+1) + u_i(i-1) + u_i(j+1) + u_i(j-1) \\ &+ u_i(k+1) + u_i(k-1)] - u_i(i+2) - u_i(i-2) \\ &- u_i(j+2) - u_i(j-2) - u_i(k+2) - u_i(k-2) \\ &- 90\, u_i\}/12\delta x^2 \end{aligned} \tag{2.29}$$

In both cases the derivatives are approximated at (i,j,k) and obvious indices are suppressed. Kwak (1975) has shown that the term $-\frac{1}{2}(D_i u_j + u_j D_j u_i)$ is conservative of both momentum and energy. This means that no momentum or energy are introduced as a result of the spatial differencing, i.e.,

$$\sum(D_i u_j + u_j D_j u_i) = 0 \quad ; \quad \sum u_i(D_i u_j + u_j D_j u_i) = 0$$

where the summation is over all grid points. The method can be called semi-conservative, because some error is introduced by the time-integration scheme.

Chapter III

THE MAIN CALCULATION

3.1 The Basic Equations

The purpose of the main calculation is to obtain, as accurately as possible, the solution of the equations of motion for homogeneous isotropic turbulence in an incompressible fluid. Physically, this flow is produced by passing a uniform stream of fluid through a mesh to produce the turbulence and then observing its decay as the turbulence proceeds downstream Special care is necessary to assure isotropy, but the experiment has been successfully carried out several times; the most recent such experiment is that of Comte-Bellot and Corrsin (1971). An alternative to grid turbulence is box turbulence, in which the fluid in a box is stirred up and allowed to return to rest. To simulate grid turbulence we will use the Navier-Stokes equations (3.1.a), together with the continuity equation for an incompressible fluid (3.1.b)·

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x_j}(u_i u_j) = -\frac{\partial p}{\partial x_i} + \nu \nabla^2 u_i \qquad (3.1.a)$$

$$\frac{\partial u_i}{\partial x_i} = 0 \qquad (3.1.b)$$

where we use the summation convention.

We shall attempt to simulate the grid turbulence experiment by selecting a cube of fluid and following its history as it passes downstream from the grid, i e., we are following it in a Lagrangian sense by invoking Taylor's hypothesis. In order to do this successfully, we must assure that the cube of fluid we select is large enough that all correlations are essentially zero at distances equal to the side of the box. From a practical point of view, this means that the box must be large compared to the integral scale of the turbulence. On the other hand, the box should also be small enough that, under the conditions of the experiment, no significant changes in important integral properties occur over a distance equal to the size of the computational cube. If these conditions

are met, we may simulate the experiment by following the time history of the cube of fluid using periodic boundary conditions in all three spatial dimensions.

Equation (3.1.b) can be replaced by an equation for the pressure p. We apply the divergence operator to Eq. (3.1.a) and note from Eq. (3.1.b) that

$$\frac{\partial}{\partial t}\frac{\partial u_i}{\partial x_i} = 0 \quad \text{and} \quad \frac{\nabla^2 \partial u_i}{\partial x_i} = 0$$

We are then left with a Poisson equation for the pressure.

$$\nabla^2 p = -\frac{\partial u_j}{\partial x_i}\frac{\partial u_i}{\partial x_j} \tag{3.2}$$

In general terms, the method of solution is to start with the velocity field at time $n\delta t$, solve Eq. (3 2) for the pressure field at time $n\delta t$, then use Eq. (3 1.a) to find $\partial u_i/\partial t$ at $n\delta t$ and advance the solution to time $(n+1)\delta t$ using the method described in the previous chapter. Thus we insure continuity at time $(n+1)\delta t$ by properly choosing the pressure at time $n\delta t$.

3.2 Derivation of the Pressure Equation

The Poisson equation (3.2) for the pressure is exact, no approximations are involved in its derivation. It is, nonetheless, instructive to examine the origin of the pressure equation from a numerical viewpoint. The final expression we arrived at in Chapter II for the velocity field at time $(n+1)\delta t$ was

$$u_i^{n+1} = u_i^n + \delta t\left[-\frac{1}{12}(u_i^*)_t^{n-1} + \frac{2}{3}(u_i^*)_t^n + \frac{5}{12}(u_i^*)_t^{n+1}\right] \tag{3.3}$$

Now we require that the numerical divergence of u_i^{n+1} be zero, i.e., $D_i u_i^{n+1} = 0$, where D_i denotes the fourth-order numerical difference approximation to $\partial/\partial x_i$, defined by Eq. (2 28).

Suppose that at this point in the calculation we have already evaluated $(u_i^*)_t^{n-1}$, $(u_i^*)_t^n$, and $(u_i^*)^{n+1}$ but not p^{*n}. We have from the Navier-Stokes equations

$$(u_i^*)_t^{n+1} = -\frac{1}{2}\left[D_j u_i^* u_j^* + u_j^* D_j u_i^*\right] + \nu D_k^2 u_i^* - D_i p^* \qquad (3.4)$$

where all missing time indices are assumed to be n+1. This leaves p^* as the only unknown. We now substitute Eq. (3.4) into Eq. (3.3), apply the numerical divergence operator to the result, and require that $D_i u_i^{n+1}$ be identically zero. This yields

$$\begin{aligned} D_i(D_i p^*) = \frac{12}{5} D_i u_i^n + D_i\Big[&-\frac{1}{5}(u_i^*)_t^{n-1} + \frac{8}{5}(u_i^*)_t^n \\ &- \frac{1}{2}(D_j u_i^* u_j^* + u_j^* D_j u_i^*) + \nu D_k^2 u_i^*\Big] \end{aligned} \qquad (3.5)$$

If Eq. (3.5) is solved exactly, the numerical divergence of u_i^{n+1} will be identically zero (within computer round-off error). The operator D_k^2 in Eq (3.5) is the fourth-order numerical difference approximation to the Laplacian defined by Eq. (2.29). The operation $D_i(D_i p^*)$ implies two sequential operations of D_i on p^*. In one dimension, D_k^2 is a five-point operator,

$$D_k^2 f(k) = [-f(k-2) + 16f(k-1) - 30f(k) + 16f(k+1) - f(k+2)]/12\delta x^2$$

and $D_i D_i$ is a nine-point operator given by

$$\begin{aligned} D_i(D_i f(i)) = [&f(i-4) - 16f(i-3) + 64f(i-2) + 16f(i-1) - 130f(i) \\ &+ 16f(i+1) + 64f(i+2) - 16f(i+3) + f(i+4)]/144\delta x^2 \end{aligned}$$

Note that this is not the simplest fourth-order approximation to $\partial^2/\partial x_i^2$, but one must use this operator (and none other) to insure that the continuity equation is satisfied exactly.

Depending on how the initial conditions are set up, the first two time steps may or may not have velocity fields whose divergences are identically zero. In our case the divergences were small but not zero. If we retain all of the terms in Eq. (3.5), including the non-zero divergences, the next two time steps will have identically zero divergences. As we explained in Chapter II, for the first few time steps of a calculation we set $(u^*)_t^n = u_t^n$ at the end of the time step for stability reasons.

All we normally have at the end of a cycle is u^n, not u_t^n. To calculate u_t^n we need the pressure field p^n, which is not normally calculated. The pressure field p^n is calculated by requiring that $D_1(u^*)_t^n = 0$. After doing this twice, we have made $D_1 u_1^{n-1}$, $D_1 u_1^n$, $D_1(u_1^*)_t^{n-1}$, and $D_1(u_i^*)_t^n$ all identically zero. By referring to Eq. (3.3), we see that the only additional requirement needed to make $D_1 u_1^{n+1} = 0$ is that $D_1(u_1^*)_t^{n+1} = 0$. To satisfy this requirement, we apply the divergence operator to Eq. (3.4) and we obtain

$$D_1(D_1 p^*) = - D_1 \left[\frac{1}{2} (D_j u_i^* u_j^* + u_j^* D_j u_1^*) \right] \tag{3.6}$$

which is considerably easier to work with than Eq. (3.5). Since the term $-(D_j u_1^* u_j^* + u_j^* D_j u_1^*)$ forms a part of u_t^*, we do the actual calculations in the following sequence.

1. Calculate $(u^*)^{n+1}$ using standard leap frog, as described in Chapter II.
2. Calculate $\left[- \frac{1}{2} (D_j u_1^* u_j^* + u_j^* D_j u_1^*) + D_k^2 u_1^* \right]^{n+1}$ and store the result in the disk file, which will later contain $(u_1^*)_t^{n+1}$.
3. Using the result from step 2, solve Eq. (3.6) for p^* (Note that the inclusion of $D_k^2 u_1^*$ does not affect this calculation, since $D_1 \left[D_k^2 u_1^* \right] = 0$.)
4. Evaluate $-D_1 p^*$ and add the result to the results of step 2, thus leaving $(u^*)_t^{n+1}$ on disk.
5. Calculate u_1^{n+1} from Eq. (3.3). We have now completed one time step.

3.3 Solution of the Pressure Equation

The Poisson equation for the pressure is solved with the help of discrete Fourier series. Any one dimensional set of N numbers which represent the values of a function f at N evenly spaced grid points $x = j\Delta x$, $j = -N/2 \ldots N/2 - 1$, can be uniquely represented by a discrete Fourier series, i.e.,

$$f(j) = \frac{1}{N} \sum_{K=-\frac{N}{2}}^{K=\frac{N}{2}-1} F(k) \exp\left(\frac{-2\pi i}{N} jk\right) \qquad (3.6.a)$$

The Fourier coefficients are given by

$$F(k) = \sum_{j=-\frac{N}{2}}^{j=\frac{N}{2}-1} f(j) \exp\left(\frac{2\pi i}{N} jk\right) \qquad (3.6.b)$$

In these equations, $f(j)$ represents the value of the function $f(x)$ at the point $x = j\delta x$. Likewise, let $p(j)$ represent the value of the unknown function $p(x)$ at $x = j\delta x$. Then the solution to the discretized Poisson equation is

$$D_x D_x p(x) = f(x) \qquad (3.7)$$

where

$$D_x f(j) = \frac{f(j-2) - 8f(j-1) + 8f(j+1) - f(j+2)}{12\delta x}$$

can be found by substituting discrete Fourier series for $p(j)$ and $f(j)$ into Eq. (3.7):

$$D_x D_x \sum_k P(k) \exp\left(\frac{-2\pi i}{N} jk\right) = \sum_k F(k) \exp\left(\frac{-2\pi i}{N} jk\right) \qquad (3.8)$$

where

$$P(k) = \sum_j p(j) \exp\left(\frac{2\pi i}{N} jk\right)$$

and

$$p(j) = \frac{1}{N} \sum_k P(k) \exp\left(\frac{-2\pi i}{N} jk\right)$$

It is easy to show that

$$D_x D_x \exp\left(\frac{-2\pi i}{N} jk\right) = g(k) \exp\left(\frac{-2\pi i}{N} jk\right) \qquad (3.9)$$

where

$$g(k) = -130 + 32\cos\left(\frac{2\pi k}{N}\right) + 128\cos\left(\frac{4\pi k}{N}\right) - 32\cos\left(\frac{6\pi k}{N}\right) + 2\cos\left(\frac{8\pi k}{N}\right)$$

Applying (3.9) to (3.8), we get from the linear independence of the complex exponentials

$$g(k)P(k)\ \exp\left(\frac{-2\pi\imath}{N}\ jk\right) = F(k)\ \exp\left(\frac{-2\pi\imath}{N}\ jk\right) \tag{3.10}$$

Now we multiply Eq. (3.10) by $\exp\left(\frac{2\pi\imath}{N}\ jk'\right)$, and, noting that for $k \neq k'$

$$\sum_{k=-\frac{N}{2}}^{k=\frac{N}{2}-1} \exp\left[\frac{-2\pi\imath}{N}\ j(k-k')\right] = 0$$

which leaves us with

$$P(k) = \frac{F(k)}{g(k)} \tag{3.11}$$

With the above in mind, we see that the equation

$$D_x D_x p(j) = f(j)$$

can be solved by the following three steps.

1. Transform $f(j)$, i.e., compute $F(k) = \sum f(j)\ \exp\left(\frac{2\pi\imath}{N}\ jk\right)$
2. Calculate $P(k) = \frac{F(k)}{g(k)}$
3. Invert $P(k)$, i.e., compute $p(j) = \frac{1}{N}\sum P(k)\ \exp\left(\frac{-2\pi\imath}{N}\ jk\right)$

The extension of the method to three dimensions is straightforward. The solution to the equation

$$\left(D_{x_1} D_{x_1} + D_{x_2} D_{x_2} + D_{x_3} D_{x_3}\right) p(j_1,j_2,j_3) = f(j_1,j_2,j_3)$$

is obtained as follows:

1. Transform $f(j_1,j_2,j_3)$, i.e , compute

$$F(k_1,k_2,k_3) = \sum_{j_1}\sum_{j_2}\sum_{j_3} f(j_1,j_2,j_3)\ \exp\left[\frac{2\pi\imath}{N}\ (j_1k_1 + j_2k_2 + j_3k_3)\right]$$

2. Calculate $P(k_1,k_2,k_3) = F(k_1,k_2,k_3)/g(k_1,k_2,k_3)$

3. Invert $P(k_1,k_2,k_3)$, i.e., compute

$$p(J_1,J_2,J_3) = \frac{1}{N^3}\sum_{k_1}\sum_{k_2}\sum_{k_3} P(k_1,k_2,k_3) \exp\left[\frac{-2\pi i}{N}(J_1k_1+J_2k_2+J_3k_3)\right]$$

The use of the fast Fourier transform (FFT) algorithm (Cochran, 1967), which requires $CN \log_2 N$ operations to perform the one-dimensional Fourier transform of N data points, makes the above method of solution practical.

3.4 Modifications to Reduce the Running Time

(a) Solving a Two-Dimensional Poisson Equation Using a One-Dimensional Fourier Transform

The two-dimensional Fourier transform of data on an $N \times N$ grid is normally accomplished by performing N one-dimensional transforms in one direction followed by N one-dimensional transforms in the second direction, for a total of $2N$ one-dimensional transforms each of length N. Since each one-dimensional transform of length N requires $CN \log_2 N$ operations, the method just described requires a total of $2CN^2 \log_2 N$ operations. The constant C represents four multiplications and two additions. Suppose we could do the same thing with a single one-dimensional transform on N^2 points. This would require $CN^2 \log_2 N^2 = 2CN^2 \log_2 N$ operations, i.e., exactly the same number as before. It turns out, however, that the machine language fast Fourier transform routine used for our problem is twice as efficient (in computing time per point) in calculating a 4096-point transform as it is in calculating a 64-point transform. Thus, if we can solve the two-dimensional Poisson equation on a 64×64 grid by using a single 4096-point transform, we can reduce the running time for that part of the problem by 50%.

In order to see how we might take advantage of this, consider the 4×4 grids illustrated in Fig. 3 1. The points in parentheses represent the virtual points which are used to provide the boundary conditions. The

		(9)	(10)	(11)	(12)		
		(13)	(14)	(15)	(16)		
(3)	(4)	1	2	3	4	(1)	(2)
(7)	(8)	5	6	7	8	(5)	(6)
(11)	(12)	9	10	11	12	(9)	(10)
(15)	(16)	13	14	15	16	(13)	(14)
		(1)	(2)	(3)	(4)		
		(5)	(6)	(7)	(8)		

Fig. 3.1.a. Normal periodic boundary condition

		(9)	(10)	(11)	(12)		
		(13)	(14)	(15)	(16)		
(15)	(16)	1	2	3	4	(5)	(6)
(3)	(4)	5	6	7	8	(9)	(10)
(7)	(8)	9	10	11	12	(13)	(14)
(11)	(12)	13	14	15	16	(1)	(2)
		(1)	(2)	(3)	(4)		
		(5)	(6)	(7)	(8)		

Fig. 3.1.b. Modified periodic boundary condition

periodic boundary conditions which are normally used are illustrated in Fig 3.1.a. The required virtual data on each line are taken from the opposite end of the same line. A modified or staggered periodic arrangement is shown in Fig. 3.1.b, here the virtual data on the horizontal lines are taken from the opposite end of the succeeding or prior line. The only difference in the two cases is that the modified conditions of Fig. 3.1.b have the left and right boundaries offset vertically by one cell. The arrangement of Fig. 3.1.b allows use of the single N^2 point transform and is therefore desirable. This raises the question of what we should require of the boundary conditions for the box turbulence problem. The first requirement is that the virtual data must have no correlation to the adjacent points. This means that the correlation of the velocity field across the box must be negligible (assuming the virtual points are taken from the opposite end of the box). This requirement is met since, as noted earlier, the box-turbulence problem must have a velocity field which is uncorrelated half way across the box if it is to make physical sense. Secondly, the row of virtual points must represent turbulence. This requirement can be met by equating the data at the virtual points to those of some other row of points in the problem. As long as the order of the points is retained (i.e., the statistics in the vertical direction are unchanged), they may be shifted vertically with no effect. Either of the boundary conditions illustrated in Fig. 3.1 satisfies the above requirements The advantage of the boundary condition illustrated in Fig. 3.1.b will soon be apparent.

Suppose we combine the four rows in Fig. 3.1 into one row of 16 points numbered $j = 1,2,3,\ldots,15,16$. For purposes of the Fourier transform, it is more convenient to use an index $m = j - 9$ so that:

$$F(k) = \sum_{m=-8}^{m=7} f(m)\ \exp\left(\frac{2\pi i}{N^2}\, mk\right) \qquad (N = 4)$$

$$f(m) = \frac{1}{N^2}\sum_{k} F(k)\ \exp\left(\frac{-2\pi i}{N^2}\, mk\right) \qquad (N = 4)$$

Now we let $j = j_1 + Nj_2$. Then $f(j_1+1,j_2) = f(j+1)$ for all points on the grid. Referring back to our derivation of $g(k)$ in the preceding section, we see that if $x = j_1\ \delta_x$,

$$D_x D_x f(m) = \frac{1}{N^2} \sum_k g_1(k)\, F(k)\, \exp\left(\frac{-2\pi i}{N^2} mk\right)$$

where

$$g_1(k) = -130 + 32\cos\left(\frac{2\pi k}{N^2}\right) + 128\cos\left(\frac{4\pi k}{N^2}\right) - 32\cos\left(\frac{6\pi k}{N^2}\right) + 2\cos\left(\frac{8\pi k}{N^2}\right)$$

$f(j_1, j_2 + 1)$ can be written $f(j+N)$, with $N = 4$ in this case, and it immediately follows that if $y = j_2 \delta y$,

$$D_y D_y f(m) = \frac{1}{N^2} \sum_k g_2(k) F(k)\, \exp\left(\frac{-2\pi i}{N^2} mk\right)$$

where

$$g_2(k) = -130 + 32\cos\left(\frac{2\pi k}{N}\right) + 128\cos\left(\frac{4\pi k}{N}\right) - 32\cos\left(\frac{6\pi k}{N}\right) + 2\cos\left(\frac{8\pi k}{N}\right)$$

Thus, wherever N^2 occurs in $g_1(k)$, it becomes N in $g_2(k)$, since a difference of one grid point in the x direction is the same as a difference of one unit in the j, but a difference of one grid point in the y direction is the same as a difference of N units in j. Hence,

$$(D_x D_x + D_y D_y) \sum_{k=-\frac{N^2}{2}}^{k=\frac{N^2}{2}-1} P(k) \exp\left(\frac{-2\pi i}{N^2} mk\right) = \sum_{k=-\frac{N^2}{2}}^{k=\frac{N^2}{2}-1} \left[g_1(k) + g_2(k)\right] \cdot P(k)\, \exp\left(\frac{-2\pi i}{N^2} mk\right)$$

and the two-dimensional Poisson equation,

$$(D_x D_x + D_y D_y)\, p(x,y) = f(x,y)$$

is equivalent to

$$(D_x D_x + D_y D_y) \sum_{k=-\frac{N^2}{2}}^{k=\frac{N^2}{2}-1} p(k)\, \exp\left(\frac{-2\pi i}{N^2} mk\right) = \sum_{k=-\frac{N^2}{2}}^{k=\frac{N^2}{2}} f(k)\, \exp\left(\frac{-2\pi i}{N^2} mk\right)$$

and can be solved by the following steps:

1 $F(k) = \sum_m f(m)\, \exp\left(\frac{2\pi i}{N^2} mk\right)$, where $m = j_1 + N j_2 - (N^2/2 + 1)$

2. $P(k) = F(k)/[g_1(k) + g_2(k)]$

3. $p(m) = \frac{1}{N^2}\sum_k P(k)\ \exp\left(\frac{-2\pi i}{N^2}\ mk\right)$

Exactly the same methods could be applied to reducing a three-dimensional transform to a one-dimensional transform, but the CDC 7600 large-core memory is not large enough to hold all of the necessary data, and the full advantage of this method is unattainable. The three-dimensional transform is therefore done by a series of modified two-dimensional transforms on each plane of data followed by a one-dimensional transform in the third direction. This reduces the running time of the Poisson solver by one-third (a 50% savings on two thirds of the transforms). Since the Poisson solver takes roughly half of the total running time, this makes the overall saving about 16% of the total.

(b) Savings from Simplified Indexing

The modified boundary conditions described in the preceding section have the additional advantage of allowing us to write all of our differencing equations in terms of one-dimensional arrays. To illustrate this, we show how a two-dimensional case can be reduced to a one-dimensional problem. Suppose we have a 64×64 array, u(64,64), and we wish to calculate its Laplacian to second-order at each point. The easiest way to program this would be to define a new array v(64,66) with $v(i,1) = u(i,64)$, $v(i,j) = u(i,j-1)$ for $j = 2, \ldots, 65$, and $v(i,66) = u(i,1)$; each of these relations holding for $i = 1, \ldots, 64$. This takes care of periodicity in the j direction, and we could then write our FORTRAN program as

```
      do 10 i=1,64
      ipl=i+1
      if(ipl.eq.65)ipl=1
      iml=i-1
      if(iml.eq.0)iml=64
      do 10 j=2,65
   10 ulap(i,j-1)=v(ipl,j)+v(iml,j)+v(i,j+1)+v(i,j-1)-4.*v(i,j)
```

There are two difficulties with the above coding Firstly, the "if" statements used to calculate ipl and iml slow the execution of the loop. With the modified periodic boundary conditions, we can simply write i-1 for iml and i+1 for ipl and we will be using the correct points even at the ends of the rows, thus eliminating the "if" statements. This reduces the running time of a typical loop by 10%. Secondly, time is required by the computer to find the address of the variable u(i,j). To find this address it must calculate m = i+64*j. Similarly, in the case of a three dimensional array, to find the address of u(i,j,k) it must calculate m=i+64*j+4096*k. With our modified boundary conditions, it is possible to code the above loop as follows.

```
      do 10 m=1,4096
   10 ulap(m)=v(m+129)+v(m+127)+v(m+192)+v(m+64)-4.*v(m+128)
```

Use of single subscripting resulted in an additional 30% savings in the typical loop. Thus, the use of staggered periodic boundary conditions allows a reduction in the running time of all the differencing calculations (i.e., virtually all the calculations other than the fast Fourier transforms) by a net 40%. In fact, it was this savings which prompted the investigation of the modified Poisson solver

The combination of the modified Poisson solver, the singly subscripted arrays, and the third-order time scheme which allowed an increased time step reduced the total running time of the main calculation by a factor of six. Whereas we initially anticipated the problem would use nine hours of computer time, the final version used only 90 minutes.

Chapter IV

TURBULENCE MODELING

4.1 The Equations of Motion

For an incompressible fluid, the Navier-Stokes equations (4.1) together with the continuity equation (4.2) describe the motion of a turbulent flow. These equations are

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x_j} u_i u_j = -\frac{\partial p}{\partial x_i} + \nu \nabla^2 u_i \tag{4.1}$$

$$\frac{\partial u_i}{\partial x_j} = 0 \tag{4.2}$$

The term $(\partial/\partial x_j)u_i u_j$ in Eq. (4.1) accounts for the change in u_i at a point in space due to the advection of momentum. The term $\nu\nabla^2 u_i$ accounts for the change in u_i at a point in space due to viscous forces.

Consider an eddy of size L whose typical velocity is U. For such an eddy, the advective term is of order U^2/L and the dissipative term is of order $\nu U/L^2$. The ratio of the advective term to the viscous term is of order UL/ν, which is the (non-dimensional) Reynolds number Re. An eddy with $Re \ll 1$ is dominated by viscous dissipation and will rapidly die out. An eddy with $Re \gg 1$ is dominated by advection and will remain in the flow for a relatively long period of time before it will die out. Hence, $Re = 1$ gives an estimate of the smallest eddy one would expect to find in the flow. We assign to the smallest eddy the size η and the velocity u_η, and since $Re = 1$ for this eddy

$$\eta u_\eta = \nu \tag{4.3}$$

In order to get another relationship between the velocity and length scales of the smallest eddy, we multiply Eq. (4.1) by u_i, which gives the equation for the kinetic energy

$$\frac{\partial}{\partial t} u_i u_i + \frac{\partial}{\partial x_j} u_j \left(\frac{u_i u_i}{2} \right) = - u_i \frac{\partial p}{\partial x_i} + u_i \nu \nabla^2 u_i \qquad (4.4)$$

Integrating Eq. (4.4) over a volume V, applying the incompressibility requirement (4.2), and ignoring contributions from the boundary yields

$$\int_V \frac{\partial}{\partial t} \frac{u_i u_i}{2} dV' = \nu \int_V u_i \nabla^2 u_i dV' \qquad (4.5)$$

The right-hand side of Eq. (4.5), when integrated by parts, is seen to be negative definite and thus represents the energy-dissipation rate. Letting ε be this energy dissipation per unit volume and noting that the dissipation occurs mainly in the eddies of size η, we have

$$\varepsilon \simeq \frac{u_\nu^2 \nu}{\eta^2} \qquad (4.6)$$

Now, since the energy dissipated by the small eddies comes from the large eddies, the dissipation rate ε is really determined by the large eddies and can be regarded as given. Then the only unknowns in Eqs. (4.3) and (4.6) are the turbulent microscales η and u_η. Solving for these we get the Kolmogoroff (1941) expressions for the turbulent microscales.

$$\eta = \left(\frac{\nu^3}{\varepsilon} \right)^{1/4} \qquad u_\eta = (\nu\varepsilon)^{1/4} \qquad (4.7)$$

A more detailed discussion of the Kolmogoroff microscales can be found in Tennekes and Lumley (1972).

In a typical problem involving turbulence, the length scale of the largest eddies which we want to simulate is several orders of magnitude larger than η. In fact, L/η, where L is the largest scale in the problem, is of order $Re_L^{3/4}$. Hence we would need $Re_L^{3/4}$ grid points in each direction, but the largest number of grid points in each dimension which one could squeeze into present-day computers for a three-dimensional calculation is of the order of 100. On the other hand, typical Reynolds numbers of engineering and scientific interest are in the range 10^4-10^8. Consequently, the grid point separation is, in general, orders of magnitude larger than η. Hence the eddies of size

η, in which the energy dissipation is occurring, cannot be calculated directly. We attempt to resolve this problem by defining a new velocity field $\bar{u}_i$ where the overbar denotes some sort of averaging process. It could denote a temporal average, a space average, or an ensemble average over many realizations. We then define u_i' by $u_i = \bar{u}_i + u_i'$.

Leonard (1973) has suggested that the appropriate averaging process for large-eddy simulation should be a local spatial average of the form

$$\bar{u}_i(\underset{\sim}{x}) = \frac{1}{V}\int_V G(\underset{\sim}{x}-\underset{\sim}{x}')u_i(\underset{\sim}{x}')dV' \tag{4.8}$$

where V is a volume surrounding the point x over which u_i is to be averaged and G is a weighting function as yet unspecified. This process may be called filtering, as its effect is to remove the small-scale fluctuations from u_i in forming $\bar{u}_i$. We call $\bar{u}_i$ the filtered or large-scale field and u_i' the subgrid scale field.

The simplest averaging operation is to let $G = 1$ and V be the cubic volume with sides of length Δ_a whose center is at $\underset{\sim}{x}$. Then

$$\bar{u}_i(\underset{\sim}{x}) = \frac{1}{\Delta_a^3}\int_{x_1-\frac{\Delta a}{2}}^{x_1+\frac{\Delta a}{2}}\int_{x_2-\frac{\Delta a}{2}}^{x_2+\frac{\Delta a}{2}}\int_{x_3-\frac{\Delta a}{2}}^{x_3+\frac{\Delta a}{2}} u_i(x_1-x_1',x_2-x_2',x_3-x_3')dx_1'dx_2'dx_3' \tag{4.9}$$

Unless otherwise noted we will take Eq. (4.9) to be the definition of $\bar{u}_i$ throughout the remainder of this dissertation.

We now take Eqs. (4.1) and (4.2) and obtain their filtered counterparts by multiplying each by the weighting function $G = 1$ and integrating over the cubic volume V to obtain

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\overline{u_i u_j} = -\frac{\partial \bar{p}}{\partial x_i} + \nu\nabla^2\bar{u}_i \tag{4.10}$$

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0 \tag{4.11}$$

We now make the substitution $u_i = \bar{u}_i + u_i'$ in the nonlinear advective term and obtain

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\overline{\bar{u}_i \bar{u}_j} = -\frac{\partial \bar{p}}{\partial x_i} + \nu \nabla^2 \bar{u}_i - \frac{\partial}{\partial x_j} \eta_{ij} \tag{4.12}$$

where

$$\eta_{ij} = \overline{\bar{u}_i u_j'} + \overline{u_i' \bar{u}_j} + \overline{u_i' u_j'} \tag{4.13}$$

is the subgrid scale counterpart of the Reynolds stress.

We stress the fact that Eq. (4.12) is exact. We have defined new variables, but so far we have made no approximations. We also point out that $\bar{u}_i$ and u_i' are continuous variables defined at all points in space and time, and they are in no way tied to the finite grid of points, which will be introduced later.

4.2 Approximations to Solve the Filtered Navier-Stokes Equations

In order to solve Eqs. (4.12) it is now necessary to make some approximations. The testing of these approximations is the purpose of our main numerical simulation. The three most common approximations used are

$$\overline{\bar{u}_i \bar{u}_j} \simeq \bar{u}_i \bar{u}_j \tag{4.14}$$

$$\overline{\bar{u}_i u_j'} + \overline{u_i' \bar{u}_j} \simeq 0 \tag{4.15}$$

$$\overline{u_i' u_j'} = f(\bar{u}_i, \bar{u}_j) \tag{4.16}$$

One of our major purposes in this work is to investigate these approximations, test their validity, and suggest improvements. Numerical tests are given in Chapter V. We give a discussion of each of these approximations below.

4.3 The Approximation (4.14) $\overline{\bar{u}_i \bar{u}_j} = \bar{u}_i \bar{u}_j$

Leonard has shown that Eq. (4.14) is probably a poor approximation in a turbulent flow. Consider a function $f(x)$ defined in some region of space. If f is fairly smooth, we can approximate $f(x)$ locally

by its Taylor series expansion about the point $\underset{\sim}{x}_o$, which is taken to be the center of the filter volume, V, over which f will be integrated to obtain $\overline{f}(\underset{\sim}{x}_o)$. Substituting the expansion of f into Eq. (4.9), we obtain.

$$\overline{f}(\underset{\sim}{x}_o) = f(\underset{\sim}{x}_o) + \frac{\Delta_a^2}{24} \frac{\partial}{\partial x_k} \frac{\partial}{\partial x_k} f(\underset{\sim}{x}) \Big|_{\underset{\sim}{x}_o} + O(\Delta_a^4) \tag{4.17}$$

where Δ_a is the length of one side of the cubic volume V. Letting $f = \overline{u}_i \overline{u}_j$, we have immediately the Leonard approximation

$$L_{ij} \triangleq \overline{\overline{u}_i \overline{u}_j} - \overline{u}_i \overline{u}_j = \frac{\Delta_a^2}{24} \frac{\partial}{\partial x_k} \frac{\partial}{\partial x_k} (\overline{u}_i \overline{u}_j) \tag{4.18}$$

The last term in Eq. (4.18) will henceforth be referred to as the Leonard term. There can be little doubt that Eq. (4.18) is a better approximation than $\overline{\overline{u}_i \overline{u}_j} = \overline{u}_i \overline{u}_j$. We now ask: What is the magnitude of the Leonard term in relation to the other terms in Eq. (4.12)?

We can get some idea of its size by considering the simple, one-dimensional Fourier wave $u = \exp(ikx)$ and the linear filter of this wave defined by

$$\overline{u}(x) = \frac{1}{\Delta_a} \int_{x-\frac{\Delta_a}{2}}^{x+\frac{\Delta_a}{2}} u(x')dx' \tag{4.19}$$

from which it is easy to show that for $u = \exp(ikx)$:

$$\overline{\overline{u}(x)\overline{u}(x)} = \frac{\sin^2\left(\frac{k\Delta_a}{2}\right)}{\left(\frac{k\Delta_a}{2}\right)^2} \frac{\sin(k\Delta_a)}{k\Delta_a} u^2(x) = \frac{\sin k\Delta_a}{k\Delta_a} \overline{u}\,\overline{u} \tag{4.20}$$

We can now quickly test the accuracy of approximation (4.18). If we approximate $\partial^2/\partial x^2$ by a second-order space-centered finite difference on a grid with spacing Δ_g and apply it to a Fourier wave, we have

$$\frac{\partial^2}{\partial x^2} (\overline{uu}) = \frac{[\cos(2k\Delta_g) - 1]}{\Delta_g^2} \overline{u}\,\overline{u} \tag{4.21}$$

We are interested in the ratio $\alpha = \overline{\bar{u}\,\tilde{u}}/\bar{u}\,\bar{u}$. The exact value is seen from Eq. (4.20) to be $(k\Delta_a)^{-1} \sin k\Delta_a$. In Table 4.1 we have given α for various values of $k\Delta_a$. The first column shows the exact result. The second column gives the approximation (4.14) for which α is always unity. The third column is the result obtained from Eq. (4.18) if exact (Fourier) differentiation is used, while the fourth column is the result obtained by using second-order finite differences, Eq. (4.21), with a grid spacing equal to $\Delta_a/2$, a value which will later be shown to be appropriate.

Table 4.1

The Leonard Term

$k\Delta_a$	Exact Eq. (4.20)	No Leonard Term (Eq. (4.14)	Fourier Eq. (4.18)	Second-Order Eq. (4.21)	Gaussian
0	1.	1.	1.	1.	1.
$\pi/4$	0.9003	1.	0.8972	0.9024	.9023
$\pi/2$	0.6366	1.	0.5888	0.6667	.6628
π	0.0000	1.	-0.6449	0.3333	.1930
$3\pi/2$	-0.2122	1.	-2.7011	0.6667	.0247
2π	0.0000	1.	-5.5791	1.0000	.0014

We see from the table that for $k\Delta_a = \pi/4$, a wave whose wavelength is eight times the averaging volume or sixteen times the grid spacing, the effect of the Leonard term is approximately ten percent and the effect increases at shorter wavelengths. Waves with $k\Delta_a > \pi$ are poorly treated by any of the approximations. However, the filter removes most of these waves (as it must to avoid the numerical problem of aliasing), so the problem is not as severe as it might seem. The oscillatory behavior of the filter at high wave number has caused some workers to replace it with a Gaussian filter. Its values are shown in the last column, and we see that Eq. (4.21) does an excellent job of matching it.

We emphasize that the error being discussed here is not related to "subgrid scale turbulence," but arises from incorrect handling of the interaction between waves which are supported by the grid.

4.4 The Approximation (4.15) $\overline{\bar{u}_i u_j'} = 0$

(a) A Model for the Cross Term

We again consider the one-dimensional wave $u(x) = \exp(ikx)$ and the filter defined by Eq. (4.19). It is easy to show that

$$\overline{\bar{u}u'} = \frac{\sin\left(\frac{k\Delta_a}{2}\right)}{\frac{k\Delta_a}{2}}\left(1 - \frac{\sin\left(\frac{k\Delta_a}{2}\right)}{\frac{k\Delta_a}{2}}\right)\frac{\sin k\Delta_a}{k\Delta_a}u^2 \tag{4.22}$$

We will now develop a model for the term $\overline{\bar{u}_i u_j'}$. From the definition of u_j' we have $u_j' = u_j - \bar{u}_j$. Using the expression (4.17) for $\bar{u}_j$,

$$u_j' \simeq -\frac{\Delta_a^2}{24}\nabla^2 u_j + O(\Delta_a^4) \tag{4.23}$$

which implies

$$\overline{\bar{u}_i u_j'} = -\frac{\Delta_a^2}{24}\overline{\bar{u}_i \nabla^2 u_j} + O(\Delta_a^4) \tag{4.24}$$

Now we substitute $u_j = \bar{u}_j - u_j'$ into Eq (4.24).

$$\overline{\bar{u}_i u_j'} = -\frac{\Delta_a^2}{24}\left(\overline{\bar{u}_i \nabla^2 \bar{u}_j} + \overline{\bar{u}_i \nabla^2 u_j'}\right) \tag{4.25}$$

The use of Eq. (4.17) to obtain Eq. (4.23) assumes that u_j is "fairly smooth". This will be true if u_j is reasonably close to $\bar{u}_j$, i.e., if u_j' does not fluctuate too rapidly. This implies that we are only modeling that portion of u_j' which is nearly resolvable on the grid and not that portion which is entirely subgrid scale.

Since $\nabla^2 u_j'$ fluctuates rapidly throughout the averaging volume and has a mean value of approximately zero, whereas $\nabla^2 \bar{u}_j$ is relatively constant throughout the averaging volume, we expect that $\overline{\bar{u}_i \nabla^2 u_j'} \ll \overline{\bar{u}_i \nabla^2 \bar{u}_j}$, and we can neglect the last term in Eq. (4 25). The lowest-order approximation to $\overline{\bar{u}_i \nabla^2 \bar{u}_j}$ is just $\bar{u}_i \nabla^2 \bar{u}_j$, so the lowest-order approximation to Eq. (4.25) is

$$C_{ij} \triangleq \overline{\bar{u}_i u_j'} = -\frac{\Delta_a^2}{24}\bar{u}_i \nabla^2 \bar{u}_j \tag{4.26}$$

Eq. (4.26) is our model for the cross term. Clearly there is no physics built into this model.

In Table 4.2 we compare the values of the cross term in the same manner as we did for the Leonard term in Table 4.1. The values given in the table are $\overline{\bar{u}\,\bar{u}'}/\overline{\bar{u}\,\bar{u}}$. In comparing the magnitudes of the Leonard and cross terms, we should recall, first, that for the values in Table 4.1 it is the difference from unity that is important, and second, that the cross term will appear with a coefficient of two in the equation. Thus we see that for $k\Delta_\Delta = \pi/4$ the cross term has approximately half the importance of the Leonard term. As a function of k, the cross term increases in size and then decreases. The approximation (4.26) for the cross term is not as accurate as the corresponding approximation for the Leonard term. These conclusions will be borne out by the results presented in Chapter V

Table 4.2

The Cross Term

$k\Delta_a$	Exact Eq. (4.22)	Fourier Eq. (4.26)	Second Order (Eq. (4.26)	Gaussian
0	0.	0.	0.	0.
$\pi/4$	0.0236	0.0257	0.0254	.0235
$\pi/2$	0.0705	0.1028	0.0976	.0718
π	0.	0.4112	0.3333	.0982
$3\pi/2$	-0.4949	0.9253	0.5690	.0376
2π	0.	1.6449	0.6667	.0058

(b) Leonard and Cross Term Energy Dissipation

Assuming that the model given by Eq. (4.18) is correct, the energy dissipation due to the Leonard term is given by

$$\varepsilon_L = -\frac{\Delta_a^2}{24}\int_V \bar{u}_i \frac{\partial}{\partial x_j}\nabla^2(\bar{u}_i\bar{u}_j)dV' \qquad (4.27)$$

Carrying through the differentiation $\partial/\partial x_j$ and noting that $\partial u_j/\partial x_j = 0$, we obtain

$$\varepsilon_L \equiv -\frac{\Delta_a^2}{24}\int_V \bar{u}_i \frac{\partial}{\partial x_k}\frac{\partial}{\partial x_k}\left(\bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j}\right) dV' \tag{4.28}$$

Now we integrate by parts twice with respect to $\partial/\partial x_k$, giving

$$\varepsilon_L = -\frac{\Delta_a^2}{24}\int_V \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j}\Delta^2\bar{u}_i dV' \tag{4.29}$$

Leonard (1973) has shown that (4.29) can be approximated in the case of homogeneous isotropic turbulence by

$$\varepsilon_L = \frac{35}{48}\Delta_a^2\left\langle\left(\frac{\partial u_1^3}{\partial x_1}\right)\right\rangle \tag{4.30}$$

For the cross-term energy dissipation we can write

$$\varepsilon_C = \frac{\Delta_a^2}{24}\int_V \bar{u}_i \frac{\partial}{\partial x_j}\left(\bar{u}_i\nabla^2\bar{u}_j\right)dV' + \frac{\Delta_a^2}{24}\int_V \bar{u}_i \frac{\partial}{\partial x_j}\left(\bar{u}_j\nabla^2\bar{u}_i\right)dV' \tag{4.31}$$

The first integral in Eq. (4.31) is identically zero, since by carrying out the differentiation $\partial/\partial x_j$ we have

$$\int_V \bar{u}_i \frac{\partial}{\partial x_j}\bar{u}_i\nabla^2\bar{u}_j dV' = \int_V \bar{u}_i\nabla^2\bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} \tag{4.32}$$

where we have again used $\partial\bar{u}_j/\partial x_j = 0$, and by integration by parts we have

$$\int_V \bar{u}_i \frac{\partial}{\partial x_j}\left(\bar{u}_i\nabla^2\bar{u}_j\right)dV' = -\int_V \bar{u}_i\nabla^2\bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} dV' \tag{4.33}$$

Since the right-hand sides of Eqs. (4.32) and (4.33) are negatives of each other,

$$\int_V \bar{u}_i \frac{\partial}{\partial x_j}\left(\bar{u}_i\nabla^2\bar{u}_j\right)dV' = 0 \tag{4.34}$$

We take the second integral in Eq. (4.31) and integrate by parts to obtain

$$\frac{\Delta_a^2}{24}\int_V \bar{u}_i \frac{\partial}{\partial x_u}\left(\bar{u}_j \nabla^2 \bar{u}_i\right) dV' = -\frac{\Delta_a^2}{24}\int_V \bar{u}_j \nabla^2 \bar{u}_i \frac{\partial u_i}{\partial x_j} dV' \qquad (4.35)$$

Comparing Eq (4.35) to Eq. (4.29) we see that the cross-term energy dissipation and the Leonard term energy dissipation are the same. Furthermore, since the skewness

$$\frac{\left\langle \left(\partial u_1/\partial x_1\right)^3 \right\rangle}{\left\langle \left(\partial u_1/\partial x_1\right)^2 \right\rangle^{3/2}}$$

is known to be negative, both terms remove energy from the flow. This will be verified by our numerical experiments.

4.5 Models for the Subgrid Scale Reynolds Stress $\overline{u_i'u_j'}$

Having developed models, for the Leonard term and the cross term, we are left with the equations

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial}{\partial x_j}(\bar{u}_i\bar{u}_j) = -\frac{\partial \hat{p}}{\partial x_i} + \nu\nabla^2\bar{u}_i - \frac{\partial}{\partial x_j}(L_{ij} + C_{ij} + \tau_{ij}) \qquad (4\ 36)$$

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0 \qquad (4.37)$$

where

$$L_{ij} = \frac{\Delta_a^2}{24}\nabla^2(\bar{u}_i\bar{u}_j) \qquad C_{ij} = -\frac{\Delta_a^2}{24}(\bar{u}_i\nabla^2\bar{u}_j + \bar{u}_j\nabla^2\bar{u}_i)$$

$$\tau_{ij} = \eta_{ij} - \frac{1}{3}\eta_{kk}\delta_{ij} \qquad \eta_{ij} = \overline{u_i'u_j'}$$

$$\hat{p} = \bar{p} + \frac{\eta_{kk}}{3}$$

We have indicated that L_{ij} and C_{ij}, while important, are really the result of interactions of the resolvable scale flow field with itself. Conversely, τ_{ij} is solely the result of the effects of the subgrid scale motions which cannot be resolved. Our only hope for modeling τ_{ij} is that the subgrid scale effects, averaged over the filter volume, are somehow functions of the resolvable part of the flow, i.e., the filtered velocity field $\bar{u}_i$. The most obvious condition that such a model must satisfy

is that, since it represents the energy transfer from the resolved large-eddy field to the small subgrid scale eddies which are dissipative,

$$-\int_V u_i \frac{\partial}{\partial x_j} \tau_{ij}\, dV' < 0$$

The simplest model which satisfies this requirement is the eddy viscosity model

$$\frac{\partial}{\partial x_j} \tau_{ij} = -\nu_T \nabla^2 \bar{u}_i$$

where the eddy viscosity ν_T is an adjustable constant. Alternatively, we can model τ_{ij} directly by

$$\tau_{ij} = -\nu_T \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) \tag{4.38}$$

where ν_T could be a constant or a function of position. The term which appears in the momentum equation is $(\partial/\partial x_j)\tau_{ij}$ which can be written

$$-\frac{\partial}{\partial x_j}\left[\nu_T \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right].$$

If ν_T is always positive, then this term can be shown to be dissipative. Since the time scale for the small-scale turbulence is much shorter than that for the resolvable scale, we expect that the small-scale eddies will adjust to the large-scale ones. It is therefore reasonable to suppose that the local subgrid scale Reynolds stress should be a function of the local level of resolvable scale turbulence. Pursuing this line of reasoning, we let ν_T in Eq. (4.38) depend on the local flow variables. We require that it be positive and have units of $(\text{length})^2 \times (\text{time})^{-1}$. The most popular such model, due to Smagorinsky (1963), uses

$$\nu_T = (c\Delta_a)^2 \left[\frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right]^{1/2} \tag{4.39}$$

Until now the only way to verify a model such as Smagorinsky's has been to observe its overall effect on the resolvable scale turbulence. With our numerical simulation we can directly compare τ_{ij} with the

results of the model at each point in space. It will be seen later that, although the Smagorinsky model of τ_{ij} is not as good as the models which have been developed for the Leonard term and the cross term, it is still reasonably accurate. We will also consider other possible models.

4.6 Combining the Leonard Term and the Cross Term

When the models for the Leonard term and the cross term are added together, we obtain

$$L_{ij} + C_{ij} = \frac{\Delta_a^2}{24} (\vec{\nabla}\overline{u}_i)\cdot(\vec{\nabla}\overline{u}_j) \qquad (4\ 40)$$

It is notable that in one dimension Eq. (4.40) reduces to

$$L_{ij} + C_{ij} = \frac{\Delta_a^2}{24} \left(\frac{\partial u}{\partial x}\right)^2 \qquad (4.41)$$

This expression is equivalent to the quadratic form of the artificial viscosity sometimes used in compressible flow calculations which was first proposed by Von Neumann (1950). The purpose of the artificial viscosity in compressible flow calculations is to smear a shock front over several cells. We note that this is precisely the effect of filtering the velocity field. If the field u has a step discontinuity, $\overline{u}$ will appear as a ramp of length $2\Delta_a$, which is exactly what the artificial viscosity attempts to do. A major difference is that we are proposing that this term be included everywhere in the calculation, whereas the traditional use of the artificial viscosity is only in regions of compression. Furthermore, the present approach provides a more rational approach to the development of this model. Since in any flow calculation one cannot resolve detail smaller than one to two meshes, we believe that this term should always be included in a calculation. Of course, in flows with relatively small gradients, its effect will be small

Chapter V

NUMERICAL RESULTS

5 1 Results of the Main Calculation

The purpose of the main calculation was to simulate a low Reynolds number turbulent flow field on a $64 \times 64 \times 64$ mesh which represents, as accurately as possible, a realization of a true turbulent flow. The computed flow can then be considered as experimental data which can be used as input for the analysis of various schemes to model the effects of turbulence. In this section we will show that the computed field is in fact a good representation of real turbulence

The experiment on which our simulation is based was reported by Comte-Bellot and Corrsin (1971). The physical experiment was the measurement of the decay of grid-generated "isotropic" turbulence in a wind tunnel A time history of the decay is obtained by employing the Taylor hypothesis This eliminates the mean flow field by assuming that the flow variables at two points in the wind tunnel separated by a distance L in the direction of the mean velocity U_o are equivalent to the flow variables at two times separated by the time $t = L/U_o$ of a flow with no mean velocity at the same point in space.

The conditions chosen to be numerically simulated are given in Table 5.1. U_o is the mean flow velocity, 10 m/sec, and M is the size of the mesh which originally generated the turbulence, 2.54 cm. The initial conditions for the numerical simulation were set up to coincide with the data at $U_o t/M = 240$ The initial conditions were given the same total energy and energy spectrum as the experimental data, and a zero divergence, but were otherwise random The data are in the form of a one-dimensional energy spectrum, $E_{11}(k)$, from which we computed the three-dimensional spectrum from the relationship (Batchelor (1953)).

$$E(k) = \frac{1}{2} k^3 \frac{2}{2k} \left[\frac{1}{k} \frac{2}{2k} E_{11}(k) \right] \qquad (5\ 1)$$

Table 5 1

Gross Properties of the Turbulent Flow

U_o = 10 m/sec, M = 2.54 cm

$\frac{U_o T}{M}$	$\sqrt{U_1^2}$ $\left(\frac{cm}{sec}\right)$	Dissipation Rate $\left(\frac{cm^2}{sec^3}\right)$	Kolmogorov Micro-Scale (cm)	Taylor Micro-Scale (cm)	R_λ $\frac{\sqrt{U_1^2}\,\lambda}{\nu}$
240	6 75	145	.069	.845	38 1
385	5.03	48.5	.091	1 09	36.6

where $E(k)$ is the three-dimensional energy spectrum. The initial field does not represent true turbulence since it contains none of the local velocity correlations that exist in a physical field. It is these correlations which give rise to the subgrid scale Reynolds stresses which we hope to model. We also note that the skewness, which is an indication of the presence of turbulence, is initially zero. The expectation is that as the equations of motion are integrated in time, a representation of a true turbulent flow will develop.

Given the fixed number of mesh points in each direction, N, the physical size of the box of fluid must be determined. The box must be large enough that the velocity correlation at $L/2$ is negligible, and it must be small enough that the highest wavenumber $k_{max} = N/L$ is large enough to include essentially all of the energy dissipation spectrum. The size of the box was chosen to be 20 cm × 20 cm × 20 cm. Fig. 5.1 gives the experimental velocity correlation function $R_{11}(r_1,0,0)$ where

$$R_{11}(r_1,0,0) = \frac{\rho_{11}(r_1,0,0)}{\rho_{11}(0,0,0)} \tag{5.2}$$

and

$$\rho_{11}(r_1,0,0) = \langle u_1(x_1,x_2,x_3)u_1(x_1+r_1,x_2,x_3)\rangle$$

We see that a 20 cm length is sufficient to meet the condition that the correlation at $L/2$ be small.

Figure 5.2 shows the three-dimensional energy spectrum $E(k)$ and the dissipation spectrum $2\nu k^2E(k)$ of the initial conditions. Since $k_{max} \simeq 10\ cm^{-1}$, we do indeed capture most of the energy and dissipation. In Fig. 5.3 we show the three-dimensional energy spectrum $E(k)$, the dissipation spectrum $D(k)$, and the energy transfer spectrum $T(k)$ of the final numerical flow field $D(k)$ is simply $2\nu k^2E(k)$. $T(k)$ is calculated from

$$\frac{\partial}{\partial t}E(k) = T(k) + D(k) \tag{5.3}$$

$\frac{\partial}{\partial t}E(k)$ was calculated using the numerical values of u_1 and $\frac{\partial}{\partial t}(u_1^*)$.

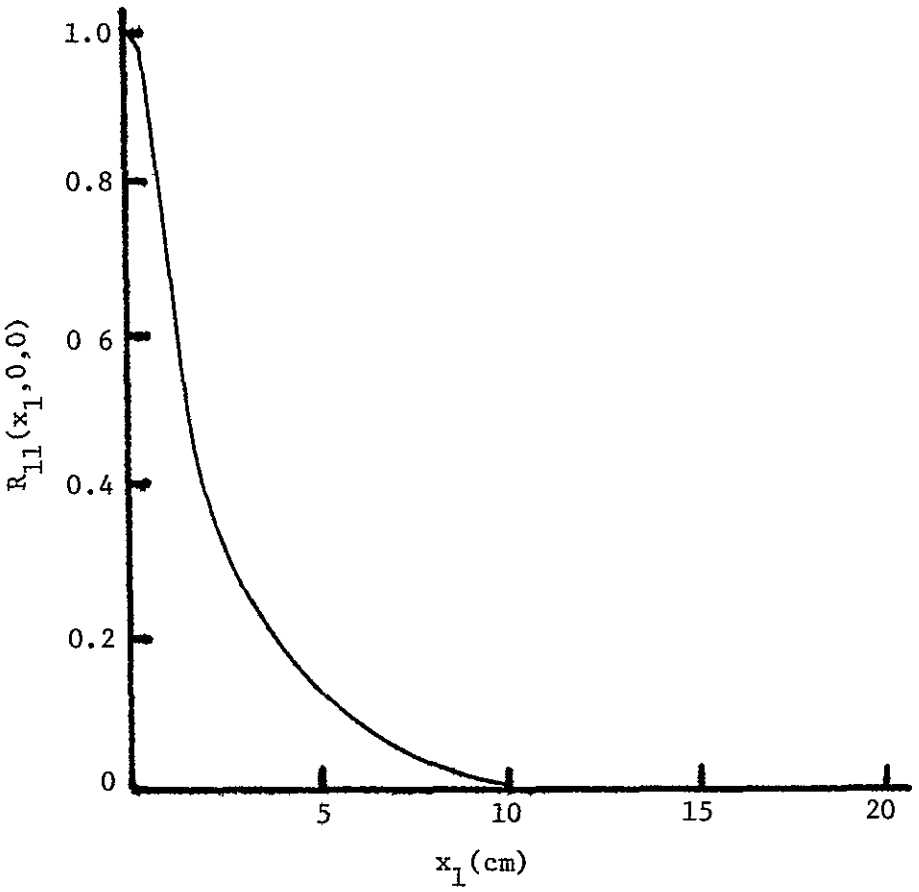


Fig. 5.1. Velocity correlation function.

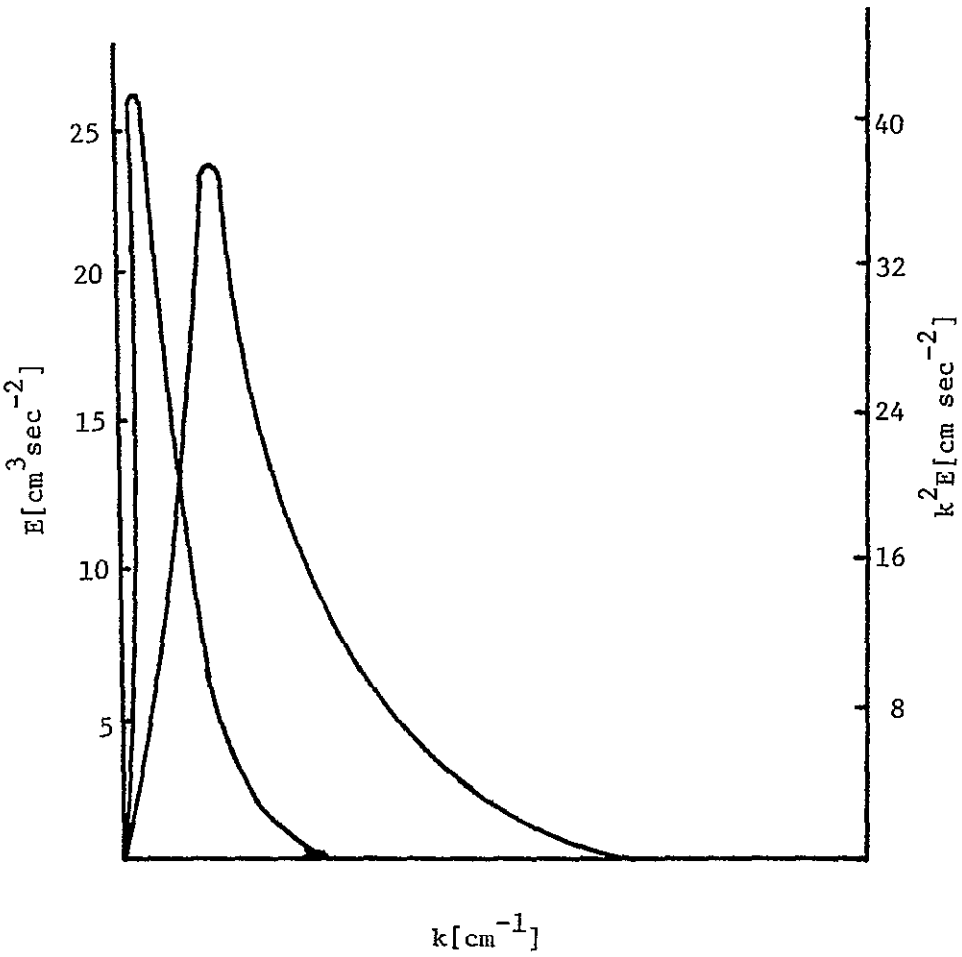


Fig. 5.2. Energy and dissipation spectrum of initial conditions.

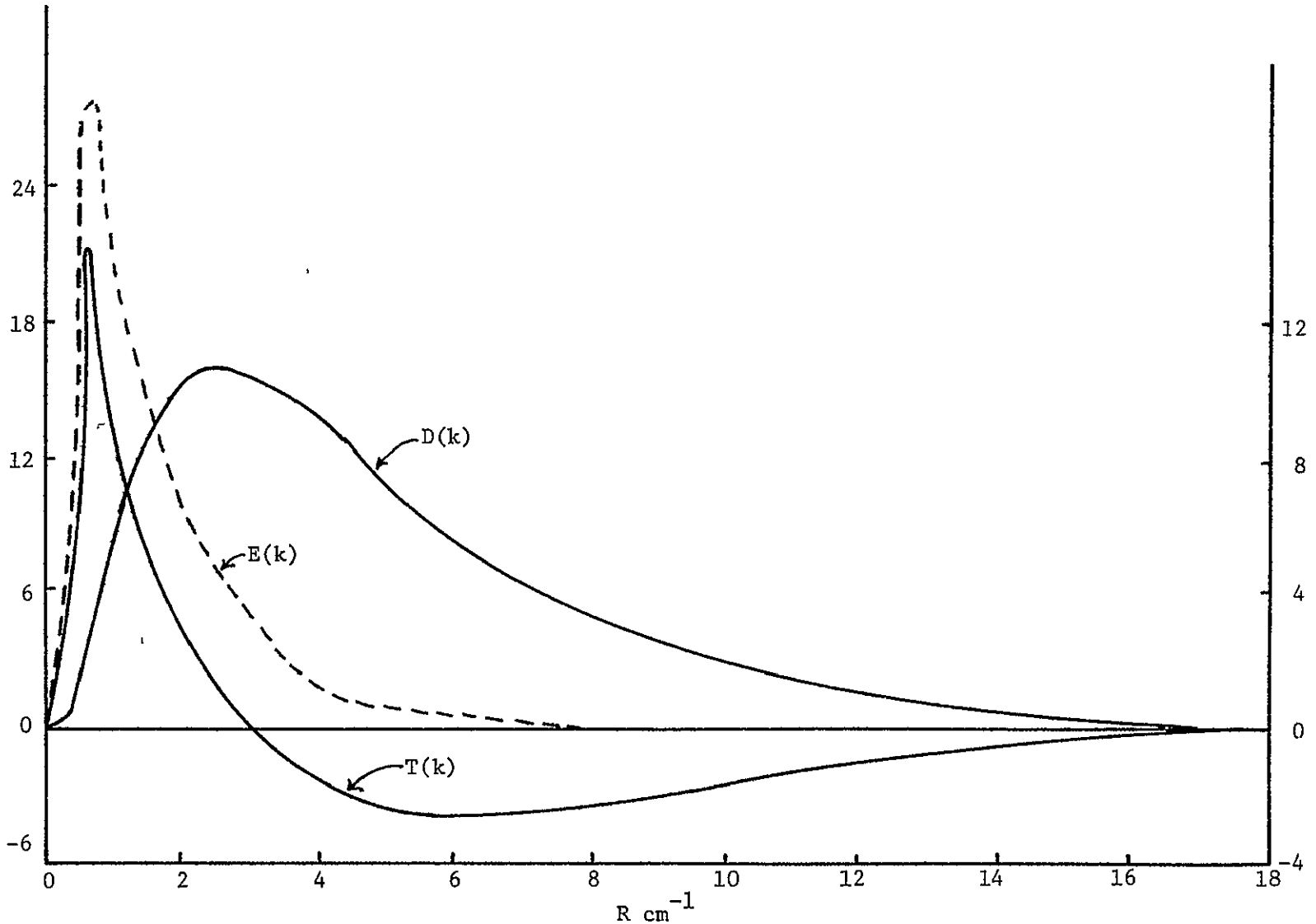


Fig. 5.3. Energy transfer and dissipation

We now examine the results of the numerical simulation. We look first at the energy, as a function of time and its rate of change, the dissipation rate We are solving the exact Navier-Stokes equations with no model for turbulent energy dissipation. The only dissipation present in the equations is a result of the physical viscosity $\nu = 0.14\ cm^2 sec^{-1}$. If we have chosen a sufficiently fine grid the energy decay of the numerical calculation will match the experimental data. As can be seen in Fig. 5.4 this is the case With $\Delta t = 0\ 0073$ seconds, 50 time steps equal 0 365 seconds, which is the elapsed time between $U_o t/M = 240$ and $U_o t/M = 385$. The total energy in the numerical simulation at time step 50 is 3 2% low. The dissipation rate, which is a more sensitive indicator, is 11 3% high. This is the result of too high a numerical transfer of energy from low to high wavenumbers. A shift to high wavenumbers will increase the dissipation, which is given by $\int \nu k^2 E(k) dk$, more than the total energy, which is given by $\int E(k) dk$.

So far, we have seen that our box is large enough to contain a sample of fluid whose velocities are uncorrelated across the box and is small enough to calculate essentially all of the real dissipation The only question remaining is, "Has the flow field developed into a truly turbulent field?" The skewness of low Reynolds number wind tunnel turbulence has been shown experimentally to be approximately -0.4 (Batchelor (1953)). The skewness S is defined as

$$S = - \frac{\left\langle \left(\frac{\partial u_1}{\partial x_1} \right)^3 \right\rangle}{\left\langle \left(\frac{\partial u_1}{\partial x_1} \right)^2 \right\rangle} \qquad (5.4)$$

where $\langle\ \rangle$ indicates an ensemble average. The skewness of the numerical flow field, with the average taken to be the average over all of the grid points, is shown in Fig. 5 5. The skewness starts at zero, since the initial flow field does not represent true turbulence. The skewness plot indicates that after only 15 time steps we appear to have stabilized the skewness. The slight dips in the skewness which occur every eight time steps were mentioned in the discussion of the third-order time-differencing scheme The third-order scheme has a weak instability which must be corrected for occasionally. In this run the correction was alternately

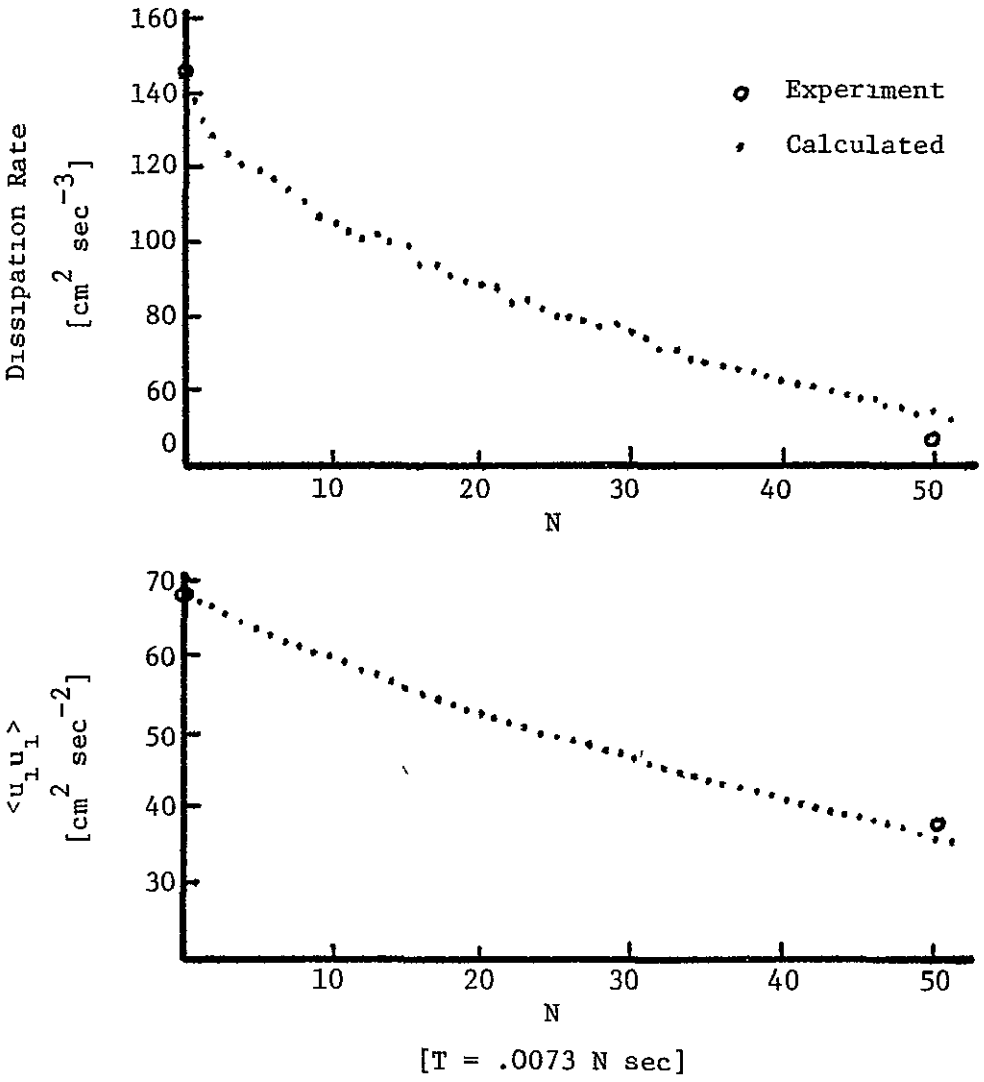


Fig. 5.4. Dissipation rate and energy

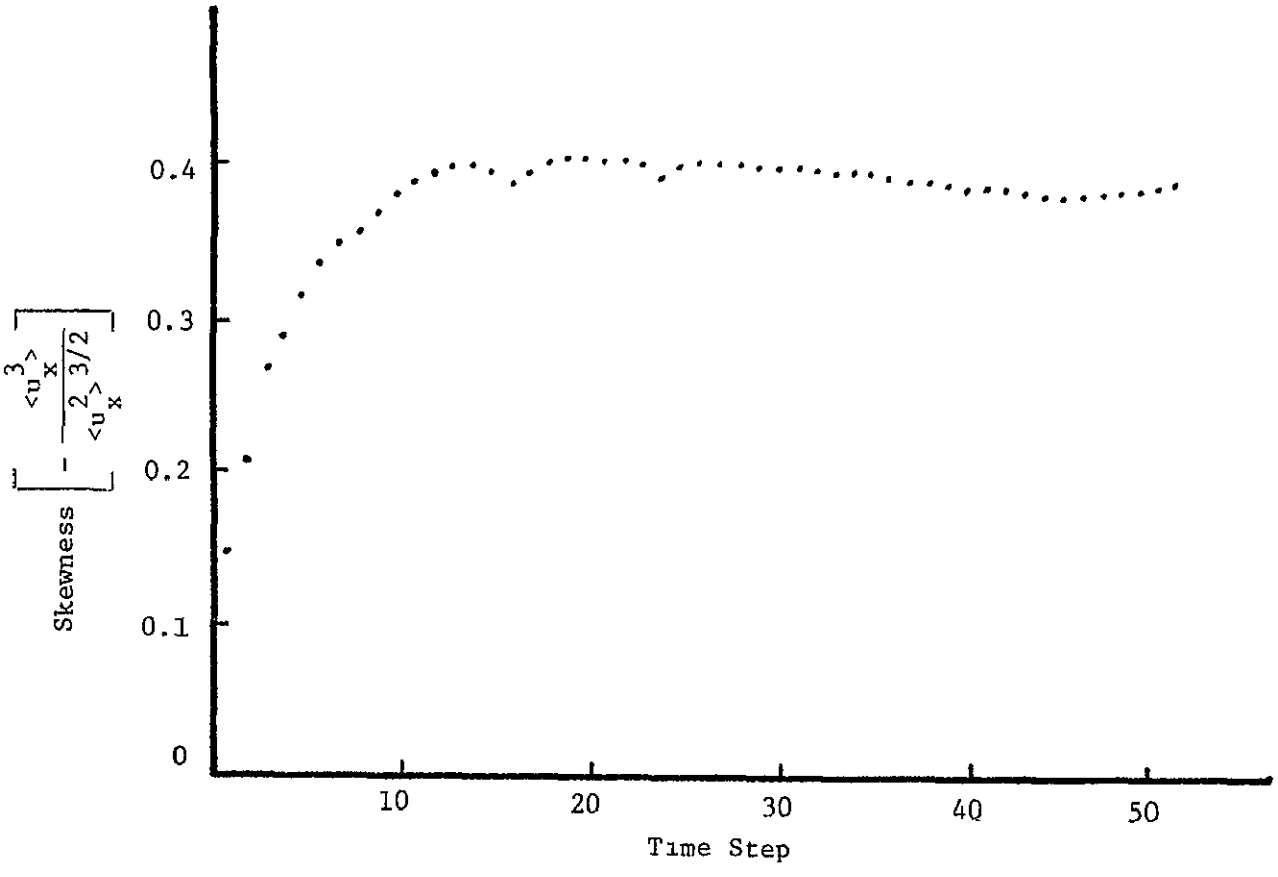


Fig. 5 5. *Skewness as a function of time*

turned on for four time steps then off for four time steps throughout the calculation. The periodic dips in the skewness coincide with the turning off of the correction. It appears that after about 30 time steps the correction was probably no longer needed.

The tailing up of the skewness near the end of the problem is probably due to the continuing accumulation of energy in the high wave numbers. Looking at the skewness, we decided that time step $n = 40$ would probably be our best representaiton of true turbulence, and this time step was chosen for the analysis to be described in the following section.

Another, much more convincing, argument that the flow is truly turbulent will be given in Section 5.4.

5.2 Testing of Subgrid Scale Modeling

Having completed the main calculation, we now have a realization of a flow field which has the characteristics of physical turbulence The data, which we treat as if it were from a physical experiment, is given on a $64 \times 64 \times 64$ mesh within a box which is 20 cm on a side. We now imagine placing a coarse $8 \times 8 \times 8$ mesh over the physical space occupied by the original fine mesh, i.e., each side of the coarse mesh is eight times a side of the original fine mesh The relation between the fine mesh and the coarse mesh is illustrated in Fig. 5.6. Within each cell of the coarse mesh we have the experimental value of the velocity field u_1 at 512 evenly spaced points. Now we need to know the value of the filtered velocity fluid $\bar{u}_1$ at each point in the fine mesh. Recall that the filtered velocity field is a continuous function defined at all points in space and is independent of the definition of the coarse mesh. We use a simple box filter with sides of length $17\Delta/8$, where Δ is the mesh spacing of the coarse grid. In order to get an average at a point, we use the value at that point and an equal number of points on either side. This means the number of points we sum over must be odd, hence $17\Delta/8$ instead of 2Δ. The value of the filtered velocity component $\bar{u}_\ell(i,j,k)$, where i,j,k are the coordinates of the point on the fine grid, is given by

$$\bar{u}_\ell(i,j,k) = \frac{1}{17^3} \sum_{i'=i-8}^{i+8} \sum_{j'=j-8}^{j+8} \sum_{k'=k-8}^{k+8} u_\ell(i',j',k') \qquad (5.5)$$

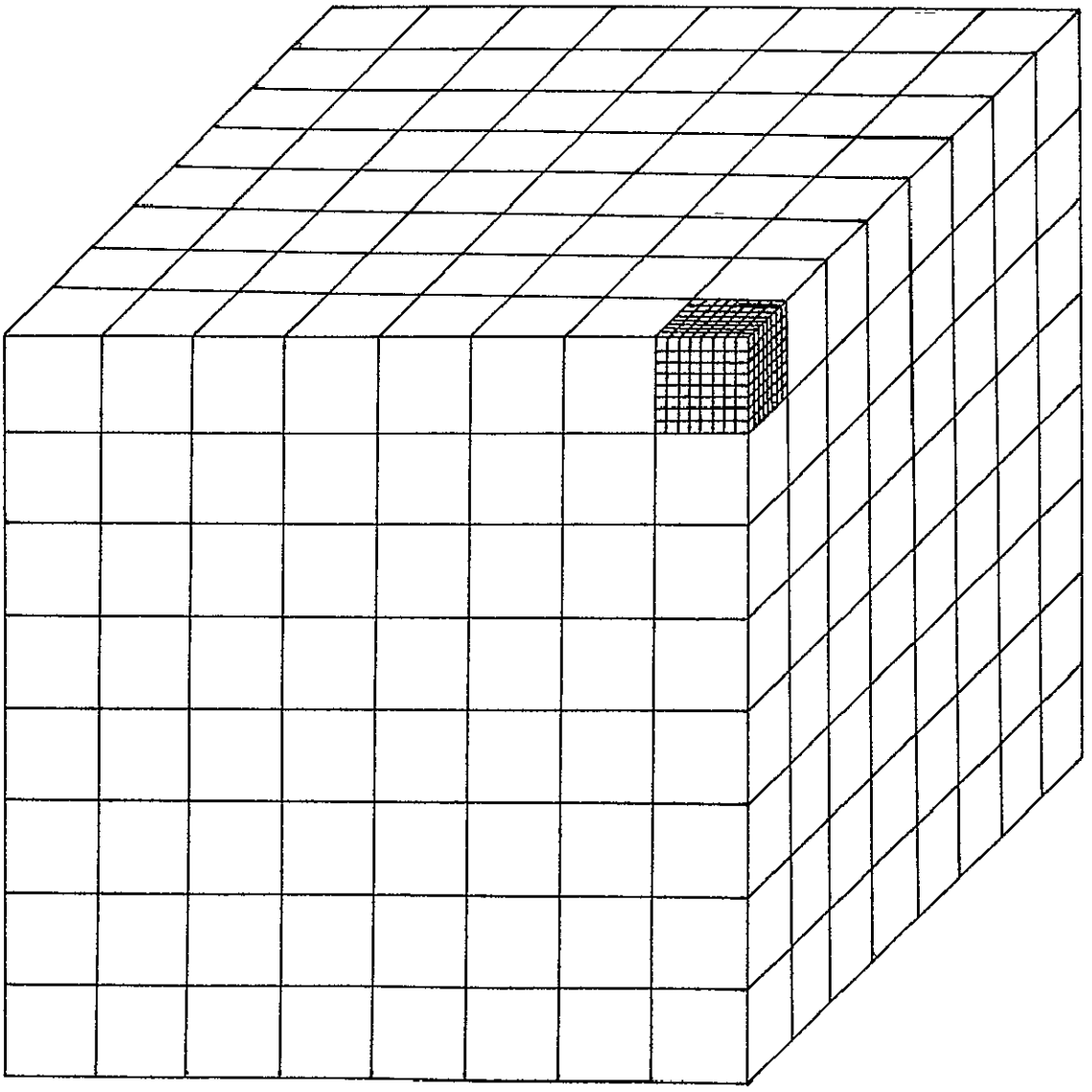


Fig. 5.6. Illustration of fine mesh inside coarse mesh

This equation is equivalent to a box filter with sides of length $17\Delta/8$ where Δ is the mesh spacing of the coarse mesh. Calculations will also be made using an averaging volume with sides of length $9\Delta/8$. Having calculated $\bar{u}_i$, we also have u_i' from its definition $u_i = \bar{u}_i + u_i'$. For illustration purposes we have randomly chosen a line of 64 points in the x_1 direction and have plotted $u_1(x_1)$ and $\bar{u}_1(x_1)$ for these 64 points in Fig. 5 7.

The remaining quantities in which we are interested are

$$\overline{\bar{u}_\ell \bar{u}_m} = \frac{1}{17^3} \sum_{i'} \sum_{j'} \sum_{k'} \bar{u}_\ell \bar{u}_m \tag{5.6a}$$

$$\overline{\bar{u}_\ell u_m'} = \frac{1}{17^3} \sum_{i'} \sum_{j'} \sum_{k'} \bar{u}_\ell u_m' \tag{5.6b}$$

$$\overline{u_\ell' u_m'} = \frac{1}{17^3} \sum_{i'} \sum_{j'} \sum_{k'} u_\ell' u_m' \tag{5.6c}$$

We now restrict our attention to the quantities $\bar{u}_i$, $\overline{\bar{u}_i \bar{u}_j}$, $\overline{\bar{u}_i u_j'}$, and $\overline{u_i' u_j'}$ at the centers of the 512 cells defined by the coarse mesh. The claim made for the turbulence models under investigation is that the variables $\overline{\bar{u}_i \bar{u}_j}$, $\overline{\bar{u}_i u_j'}$, and $\overline{u_i' u_j'}$ can be expressed as functionals of $\bar{u}_i$ We will now demonstrate the extent to which this is true for the case of low Reynolds number, isotropic, homogeneous turbulence of an incompressible fluid.

5.3 The Energy Dissipation

The equation for the kinetic energy of the filtered velocity field may be obtained by taking the scalar product of Eq. (4.3b) with $\bar{u}_i$ We obtain

$$\frac{1}{2} \frac{\partial}{\partial t} (\bar{u}_i \bar{u}_i) + \bar{u}_i \frac{\partial}{\partial x_j} (\bar{u}_i \bar{u}_j) = - \bar{u}_i \frac{\partial \hat{p}}{\partial x_i} + \nu \nabla^2 \bar{u}_i + \bar{u}_i \frac{\partial}{\partial x_j} (L_{ij} + C_{ij} + \tau_{ij}) \tag{5.7}$$

For our purposes the dissipation due to the real viscosity can be ignored since we find that it represents less than five percent of the total dissipation on the coarse grid. When Eq (5.2) is integrated over all space

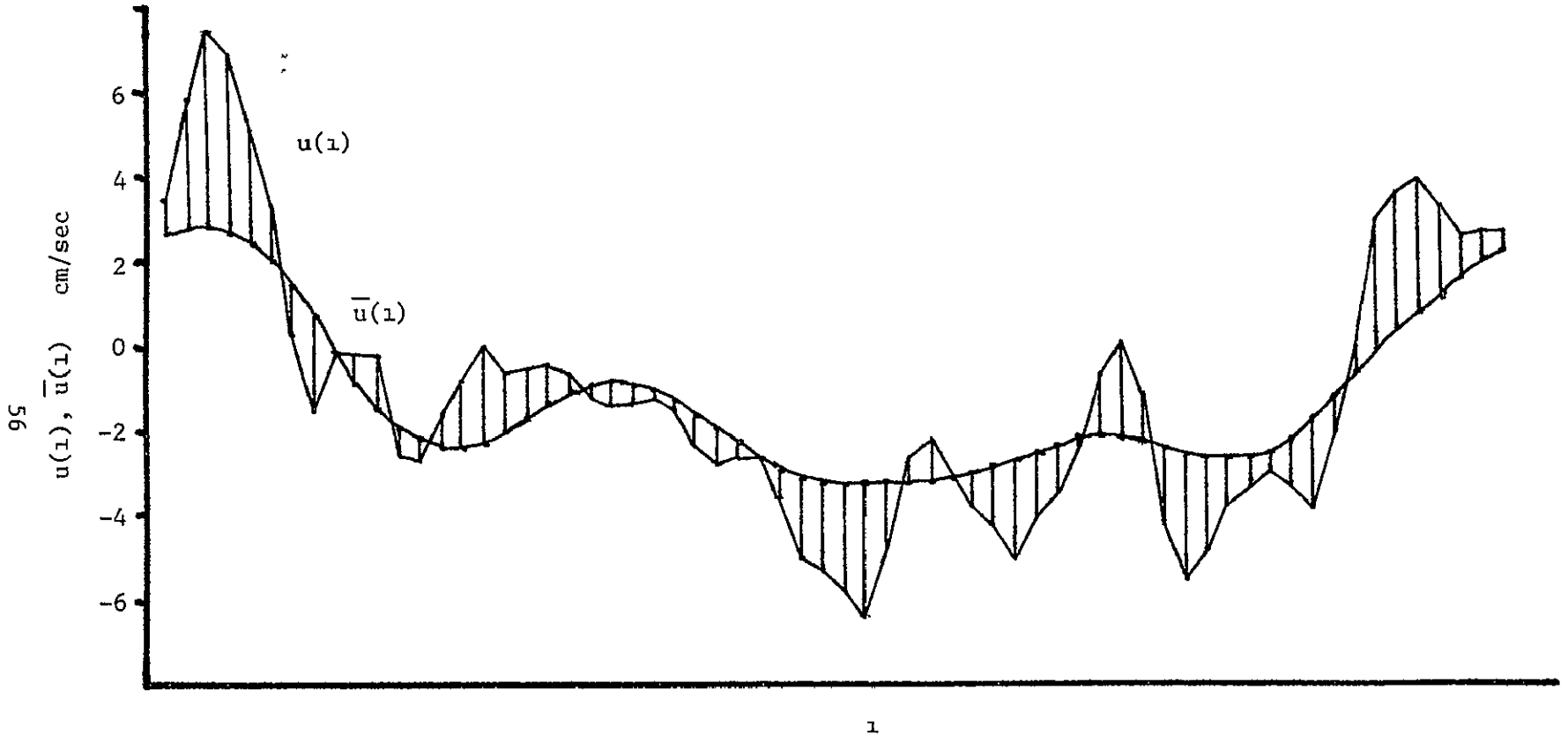


Fig. 5.7. Sample of filtered and unfiltered velocity fields

the contributions from the pressure gradient and the nonlinear advective term vanish, leaving

$$\frac{1}{2}\int_V \frac{\partial}{\partial t}(\bar{u}_i\bar{u}_i)dV' = -\int_V \bar{u}_i \frac{\partial}{\partial x_j} L_{ij}dV' - \int_V u_i \frac{\partial}{\partial x_j} C_{ij}dV' - \int_V \bar{u}_i \frac{\partial}{\partial x_j} \tau_{ij}dV' \quad (5.8)$$

Using our experimental values of L_{ij}, C_{ij}, and τ_{ij} for each point in the coarse mesh, we can calculate the dissipation rate due to each term.

$$\varepsilon_L = \frac{1}{512}\sum_{n=1}^{512} -\bar{u}_i \frac{\partial}{\partial x_j} L_{ij} \quad (5.9a)$$

$$\varepsilon_C = \frac{1}{512}\sum_{n=1}^{512} -\bar{u}_i \frac{\partial}{\partial x_j} C_{ij} \quad (5.9b)$$

$$\varepsilon_\tau = \frac{1}{512}\sum_{n=1}^{512} -\bar{u}_i \frac{\partial}{\partial x_j} \tau_{ij} \quad (5.9c)$$

The results are plotted in Fig. 5.8 for ε_τ at four time steps, using the averaging volume of $9\Delta/8$ on a side. Some comments on Fig. 5.8 are in order. First, we note that at time step $n = 1$, i.e., the initial condition, the dissipation due to the subgrid scale Reynolds stresses is zero. This is what we expect, since the initial conditions do not represent true turbulence. As the flow develops and becomes more physical, the dissipation term from the subgrid scale Reynolds stress rises rapidly before falling off as the energy of the flow decreases. This is the evidence referred to in Section 5.2 which indicates the flow has developed a truly turbulent nature. The finite differences used to obtain Fig 5.8 were taken on the coarse mesh to conserve computing time. It is more accurate to take differences of the filtered quantities on the fine mesh (recall that the filtered quantities are defined at all points in space), and this was done at the final time step to obtain the dissipation from the cross and Leonard terms as well as the subgrid scale Reynolds stress. At time step 41 the dissipation rate due to the subgrid scale Reynolds stresses found in this way is 5.32 cm^2/sec^3, which is a bit less than that found by coarse mesh differencing, that from the

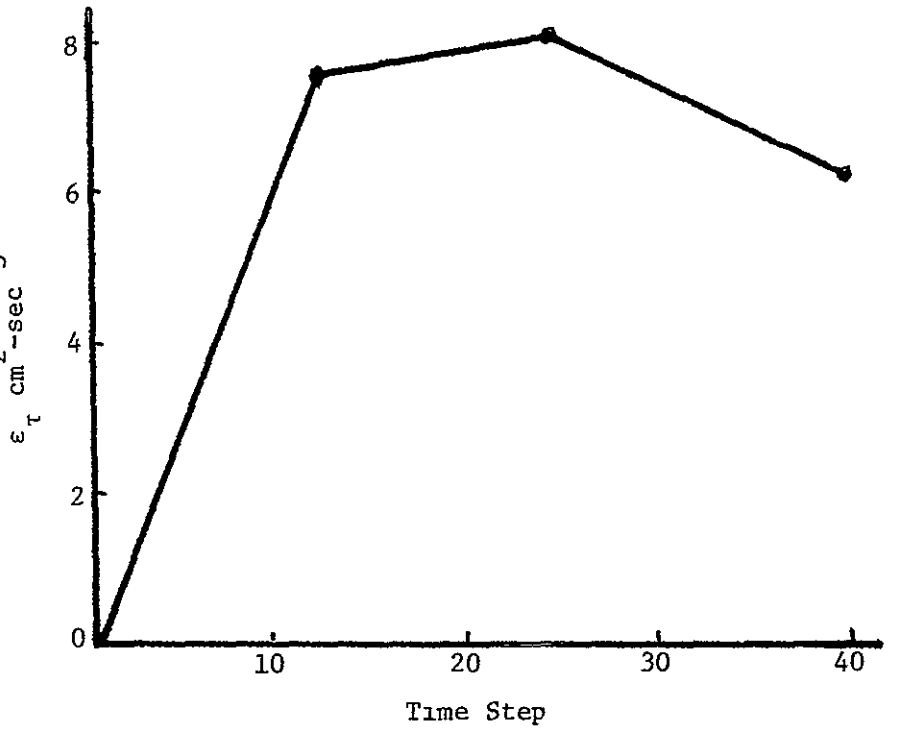


Fig. 5.8. Dissipation rate ε_τ

cross term is 3.95 cm^2/sec^3, and that from the Leonard term is 2 82 cm^2/sec^3. We recall that the models predicted that the dissipation from the cross and Leonard terms should be equal. Though not equal, they are reasonably close.

The dissipation for the large $(17\Delta/8)$ filter case was also calculated at time step $n = 41$. The major difference from the previous case is a decrease in the dissipation due to the cross and Leonard terms. This is because increasing the size of the filter smooths the resultant field, causing a decrease in skewness and hence a decrease in cross-term dissipation.

5.4 Correlations Between the Models and the Numerical Experiment

We are now in the position of being able to make direct comparisons between the models for subgrid-scale turbulence terms and their numerical experimental values. We define the correlation coefficient $C(M,X)$ between the values of the model M and the experiment X as

$$C(M,X) = \frac{<MX>}{<M^2>^{1/2} <X^2>^{1/2}} \tag{5.10}$$

where

$$<MX> = \frac{1}{512} \sum_{n=1}^{512} M(n)X(n) \tag{5.11a}$$

$$<M^2> = \frac{1}{512} \sum_{n=1}^{512} M^2(n) \tag{5 11b}$$

$$<X^2> = \frac{1}{512} \sum_{n=1}^{512} X^2(n) \tag{5.11c}$$

If M and X are totally unrelated, then $C(M,X) = 0$ If M is a constant multiple of X, i.e., if the model is exact, $C(M,X) = 1$.

There are three levels at which comparisons can be made, and these correspond to how the terms appear in the equations For the moment we restrict ourselves to discussing the subgrid-scale Reynolds stress τ_{ij} The most direct comparison is at the tensor level, i.e , between the

experimental and modeled values of τ_{ij}. However, the term which actually occurs in the momentum equation is an acceleration vector $\partial\tau_{ij}/\partial x_j$. We define the vector level of comparison to be that between the experimental and modeled values of $\partial\tau_{ij}/\partial x_j$. The scalar level of comparison refers to the energy dissipation $\bar{u}_i(\partial\tau_{ij}/\partial x_j)$ produced at each cell in the coarse mesh by the experimental τ_{ij} and the modeled τ_{ij}. The primary purpose of the subgrid-scale model is to remove kinetic energy at the correct portion of the flow, so the scalar level of comparison is important, and we will find it to be considerably better than the other two

5.5 Tensor-Level Comparisons

(a) The Leonard Term

The Leonard term is defined as

$$L_{ij} = \eta_{ij} - \frac{1}{3}\eta_{kk}\delta_{ij} \quad ; \quad \eta_{ij} = \overline{\bar{u}_i\bar{u}_j} - \bar{u}_i\bar{u}_j \qquad (5.12)$$

and the model we use is

$$\eta_{ij} = \alpha_{ij} - \frac{1}{3}\alpha_{kk}\delta_{ij} \quad ; \quad \alpha_{ij} = \frac{\Delta_a^2}{24}\nabla^2(\bar{u}_i\bar{u}_j) \qquad (15.13)$$

Fourth-order differencing has been used in evaluating all of the models which we will be discussing. Fourth-order space differencing gives one to three percent better correlations than second-order differencing. The differencing was done on the coarse mesh, because this is the mesh which would be available in a real simulation. We note that we can do better in this case, since we have $\bar{u}_i$ on the fine mesh as well as on the coarse mesh and can get a better approximation of the actual derivatives. We compared the results obtained by differencing on the fine mesh to those obtained on the coarse mesh and found the differences to be minor (the correlations for the Leonard, cross, and Reynolds terms in the case of the small filter changed from 0.909, 0.790, and 0.277 to 0.934, 0.744, and 0.297, respectively).

We find that the correlation between the model (5 13) and the experiment (5.12) is 0.935 for the large (17Δ/8) filter and 0.909 for the small (9Δ/8) filter. The ratios of the r.m.s. value of the model to the r.m.s. value of the experiment is 1.60 for the large filter and 0.788 for the small filter; the reason why these values differ from each other and from unity are not understood. Since the model for the Leonard term involves the fewest approximations of the three terms we are considering, we expect it to be the best, which it is. Also, since the model is based on a Taylor series expansion of the filtered velocity field, we expect the smoother velocity field produced by the larger filter to give better results, and it does. The correlations as functions of i,j are detailed in Table 5.2. Only small differences are observed throughout the table.

(b) The Cross Term

The cross term is defined as

$$C_{ij} = \eta_{ij} - \frac{1}{3}\eta_{kk}\delta_{ij} \quad ; \quad \eta_{ij} = \overline{\bar{u}_i u'_j} + \overline{u'_i \bar{u}_j} \tag{5.14}$$

and the model we use is

$$M_{ij} = \alpha_{ij} - \frac{1}{3}\alpha_{kk}\delta_{ij} \quad ; \quad \alpha_{ij} = -\frac{\Delta_a^2}{24}\left(\bar{u}_i \overline{\nabla^2 u_j} + \bar{u}_j \overline{\nabla^2 u_i}\right) \tag{5 15}$$

In this case we find that the correlation is better for the smaller filter than for the larger filter. This is probably because the experimental values are smaller for the large filter than for the small filter, due to the smoother flow field. The correlations of 0.685 and 0.790 are less than for the Leonard term, but the r.m.s. ratios of model to experiment are better, i.e., 1.23 and 0.96. Details are in Table 5.3.

(c) The Subgrid-Scale Reynolds Stress

The definition of the subgrid-scale Reynolds stress is

$$\tau_{ij} = \eta_{ij} - \frac{1}{3}\eta_{kk}\delta_{ij} \quad ; \quad \eta_{ij} = \overline{u'_i u'_j} \tag{5.16}$$

Table 5.2

Correlation Between Exact and Modeled Leonard Terms

$\Delta_A = \frac{17\Delta}{8}$

J \ i	1	2	3
1	.92	.94	.94
2	.94	.93	.93
3	.94	.93	.94

Average = .935

$$\frac{< M^2 >^{1/2}}{< L^2 >^{1/2}} = 1.60$$

$\Delta_A = \frac{9\Delta}{8}$

J \ i	1	2	3
1	90	.91	.93
2	91	.89	.92
3	.93	.92	.90

Average = .909

$$\frac{< M^2 >^{1/2}}{< L^2 >^{1/2}} = .788$$

Table 5.3

Correlation Between Exact and Modeled Cross Term

J \ I	1	2	3
1	.69	.67	.72
2	67	.64	.70
3	72	.70	.65

Average = .685

$$\frac{\langle M^2 \rangle^{1/2}}{\langle C^2 \rangle^{1/2}} = 1.23$$

J \ I	1	2	3
1	.78	76	82
2	.76	78	80
3	82	.80	.78

Average = .790

$$\frac{\langle M^2 \rangle^{1/2}}{\langle C^2 \rangle^{1/2}} = .96$$

The four models we use are

$$M_{ij} = \alpha_{ij} - \frac{1}{3}\alpha_{kk}\delta_{ij} \quad , \qquad \alpha_{ij} = K\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) \tag{5.17}$$

where K is given by

$$\text{model 1} \qquad K = (C\Delta_a)^2 \left[\frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right)\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right)\right]^{1/2} \tag{5.18a}$$

$$\text{model 2} \qquad K = (C\Delta_a)^2 \,(\bar{\omega}_i \bar{\omega}_i)^{1/2} \tag{5.18b}$$

$$\text{model 3} \qquad K = (C\Delta_a)\,\frac{1}{3}\,(\overline{u_k' u_k'})^{1/2} \tag{5.18c}$$

$$\text{model 4} \qquad K = C \tag{5.18d}$$

In Eq. (5.18b), ω_i represents the vorticity $\omega_i = \varepsilon_{ijk}(\partial u_k/\partial x_j)$ All four models were found to be equally valid with the best correlations found to be 0.363 for the large filter and 0.303 for the small filter. In model 3 the value for $\overline{u_k' u_k'}$ is taken from the experimental data. Although these correlations are considerably below those for the cross term and the Leonard term, they are clearly significant. The constants in the models were obtained by matching the r.m.s. value of the exact quantity with that of the model prediction. Detailed comparisons for the Smagorinsky model are shown in Table 5 4, and a summary for all models is given in Table 5 5

5.6 Vector-Level Comparison

(a) The Leonard and Cross Terms

In the previous section we compared the models directly with the corresponding stress tensors. Here we make the comparison with the terms which actually enter the momentum equations, i.e.,

$$\frac{\partial}{\partial x_j} L_{ij} \quad ; \qquad \frac{\partial}{\partial x_j} C_{ij}$$

Table 5.4

Correlation of Subgrid-Scale Reynolds Stresses with Smagorinsky Model

J \ I	1	2	3
1	.20	51	.38
2	.51	.23	.39
3	.38	39	.26

Average = .346

C = .269

J \ I	1	2	3
1	18	.41	.32
2	.41	21	.28
3	.32	.28	.26

Average = 277

C = .247

Table 5 5

Summary of Correlations between Exact Subgrid Scale Reynolds Stresses and Models

Term	Model	$\frac{17}{8}\Delta$	$\frac{9}{8}\Delta$	$\frac{17}{8}\Delta$	$\frac{9}{8}\Delta$
R_{ij}	Smagorinsky (5.18a)	.346	.277	270	.247
	Vorticity (5.18b)	.344	.260	.294	.275
	T.K.E. (5.18c)	.363	.303	.196	.175
	Eddy viscosity (5.18d)	.352	.295		
$\frac{\partial R_{ij}}{\partial x_j}$	Smagorinsky	.425	.346	.240	.264
	Vorticity	.408	.327	.220	.247
	T.K.E.	.434	.362	.138	.155
	Eddy viscosity	.426	.356		
$u_i \frac{\partial R_{ij}}{\partial x_j}$	Smagorinsky	.710	.580	.186	.171
	Vorticity	700	.582	.202	.191
	T.K.E.	.723	.606	.085	.095
	Eddy viscosity	716	.605		

The results for the large filter show that the correlations range from 0.935 to 0.947 for the Leonard term and from 0.685 to 0.689 for the cross term. For all practical purposes these are the same as for the direct comparison.

(b) The Subgrid-Scale Reynolds Stress

In contrast to the case for the Leonard and cross terms, we find a significant increase in the correlations between $\partial\tau_{ij}/\partial x_j$ and its experimental value over the direct correlation between the stress tensors. The results shown in Table 5.5 show that all models are again equally good, but the correlation has typically increased from 0.35 to 0.42 for the large filter. Comparable increases are seen in the small filter results. The reason for this increase is not understood. We note also that the model constants decrease, with one exception; again the reason is not understood.

5.7 Scalar-Level Comparison

(a) The Leonard and Cross Terms

Here we make our comparisons based on the terms which enter the energy equation, i.e., $\overline{u}_i\,(\partial L_{ij}/\partial x_j)$ and $\overline{u}_i\,(\partial C_{ij}/\partial x_j)$. Summaries for the three levels of comparison are given in Table 5 . We see a small decrease in the correlations from the vector to the scalar level for both the Leonard and the cross terms. The relatively large disagreement in the magnitude of the dissipation due to the Leonard term and model are not considered serious, since the dissipation due to the Leonard term is relatively small

(b) The Subgrid-Scale Reynolds Stress

We see a very sharp increase in the correlations for the subgrid-scale Reynolds stress at the scalar level. For example, at the vector level the Smagorinsky model with $\Delta_a = 17\Delta/8$ had a correlation of 0.425, but at the scalar level it is 0.710. Part of the increase may be due to the fact that both the experimental and modeled terms have mean values which are significantly positive. Even so, when the mean values of both

are subtracted out, the correlation between the fluctuating components of the exact and model values is still 0.535. We also note a further decrease in the model constant.

5.8 The Subgrid-Scale Eddy Coefficient

The models (5.18) contain constants which are usually called the subgrid-scale eddy coefficient. The value of the constant has no effect on the correlation between model and experiment As mentioned above, we can, however, adjust the constant to match the r m.s. values of the model to experiment. The values of the constants found in this way are given in Table 5.5 and were mentioned earlier. The constants obtained decrease as we pass from the tensor level of comparison to the scalar level. Since the primary function of these models is to represent the transfer of energy from large to small scales, which acts like a dissipation in the large scales, we recommend that the values given for the scalar level of comparison be used. For the Smagorinsky model, these values are in excellent agreement with theoretical and experimental values which range from around 0.13 to 0 21 (Deardorff (1971)). We note that when the Smagorinsky model is formulated using the term $(C\Delta_a)^2$, the value of C is nearly independent of Δ_a; this would not be the case if the grid spacing Δ_g were used. It is encouraging that we have obtained about the same value for C as is obtained by theoretical arguments assuming an inertial sub-range and by numerical experiments at high Reynolds numbers, even though we are at low Reynolds number and have no discernible inertial range This leads us to speculate that C is relatively independent of the spectrum of turbulence, at least in the isotropic case The values we obtain are within ten percent of those found by Kwak et al. (1975) and Shaanan et al. (1975) by matching model calculations to experimental energy decay. Since a change in numerical method can result in a ten percent change in the constant, we can say that we have indeed predicted the model constant without reference to experiment

5.9 Comments on the Correlations

A striking result that can be reached from looking at Table 5.5 is that all four of the proposed models are essentially equally valid. Since all of the models use a positive scalar times $(\partial\bar{u}_i/\partial x_j + \partial\bar{u}_j/\partial x_i)$, we checked to see how often the sign of τ_{ij} coincided with the sign of $(\partial\bar{u}_i/\partial x_j + \partial\bar{u}_j/\partial x_i)$ and found it to be only 68%. We also ran a calculation with K being arbitrarily adjusted at each point in space so as to give the best possible correlation. At the tensor level of comparison, we achieved a correlation of 0.51, versus numbers around 0.35 for the models considered above. We conclude that no model of the form (5.17) can do significantly better than Smagorinsky's. This includes models which attempt to calculate the transport of turbulent kinetic energy and models which attempt to calculate both the turbulent kinetic energy and a length scale or the dissipation (so-called two-equation models). This is partially verified by the results of model (5.18c), which show that even if one could calculate exactly the turbulent kinetic energy in each cell this would not give a significant improvement.

We also include in Table 5.5, at the vector level, a modified Smagorinsky model where, instead of

$$\frac{\partial}{\partial x_j}\left[K\left(\frac{\partial\bar{u}_i}{\partial x_j} + \frac{\partial\bar{u}_j}{\partial x_i}\right)\right] ,$$

we used

$$K\nabla^2\bar{u}_i \qquad (5\ 19)$$

The correlation decreased only slightly. The Smagorinsky model has the disadvantage that its finite-difference form does not detect a wave with $k = \pi/\Delta$, i.e , a sawtooth, since its first derivative is always calculated as zero. This can result in the failure to dissipate sufficient energy at high wave numbers. The modified Smagorinsky model does detect and dissipate these waves by using the finite-difference approximation to ∇^2. Model (5.19) has the disadvantage that one cannot rigorously prove that it is dissipative

5.10 Other Models, Which Were Discarded

All of the models considered above are reasonably good We list here some of the more reasonable-looking models which were tried but discarded. The following three tensor eddy viscosity models all had correlations of less than 0.02 with the numerical experiment.

$$\tau_{ij} = C\Delta_a^2 D_{ik}D_{kj} \tag{5.20a}$$

$$\tau_{ij} = C\Delta_a^2 \frac{1}{2}(R_{ik}D_{kj} + R_{jk}D_{ki}) \tag{5.20b}$$

$$\tau_{ij} = C\Delta_a^2 \frac{1}{2}(D_{ik}D_{kj} + R_{jk}R_{ki}) \tag{5.20c}$$

where D_{ij} is the strain rate tensor,

$$D_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right)$$

and R_{ij} is the rotation tensor,

$$R_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} - \frac{\partial \bar{u}_j}{\partial x_i}\right)$$

The next three models were proposed because of their similarity to the Leonard term, and all had correlations with τ_{ij} of less than 0.02.

$$\tau_{ij} = \frac{\partial}{\partial x_k}\frac{\partial}{\partial x_k}\bar{u}_i\bar{u}_j \tag{5 21a}$$

$$\tau_{ij} \quad \frac{\partial}{\partial x_i}\frac{\partial}{\partial x_j}\bar{u}_k\bar{u}_k \tag{5.21b}$$

$$\tau_{ij} = \frac{1}{2}\left[\frac{\partial}{\partial x_i}\frac{\partial}{\partial x_k}(\bar{u}_j\bar{u}_k) + \frac{\partial}{\partial x_j}\frac{\partial}{\partial x_k}(\bar{u}_i\bar{u}_k)\right] \tag{5 21c}$$

Chapter VI

CONCLUSIONS AND RECOMMENDATIONS

Most of the following conclusions are strictly valid only in the case of low Reynolds number homogeneous isotropic turbulence. For some of them, the range of validity extends beyond the range for which it has been proven; for others the validity of such extensions is unclear.

Our first conclusions and recommendations are concerned with the simulation itself.

1. With the present computer capacity it is possible to simulate homogeneous isotropic turbulence accurately in three dimensions. The limitation to a $64 \times 64 \times 64$ grid restricts the Reynolds number based on Taylor microscale to $R_\lambda \lesssim 40$.

2. The use of the third-order time method that we have developed allows a considerably greater time step to be used with very little sacrifice in computational time or accuracy. We recommend the use of third- or fourth-order methods in future simulations, and some work should be done to find the optimum such method.

3 The use of staggered periodic boundary conditions allows a considerable increase in computational efficiency at no cost whatsoever.

4. The results of our simulation agree with the results of the corresponding experiment in all significant statistical quantities, and we are confident that they may be used for model testing.

5. With larger computers that will be available in a short time, it will be possible to use $256 \times 256 \times 256$ grids and increase the Reynolds numbers considered by a factor of four. We believe that these computations are important and should be done

The next set of conclusions and recommendations is concerned with the models used to represent the subgrid scale turbulence.

6. The Leonard term is indeed of considerable importance in the prediction of turbulent flows and should be included in any simulation. The approximation to $(\overline{\bar{u}_i \bar{u}_j} - \bar{u}_i \bar{u}_j)$ suggested by Leonard is fairly accurate, although some adjustment of the constant may be desirable. An alternative is direct computation of the terms.

7. The cross term, $(\overline{\bar{u}_i \bar{u}'_j} + \overline{\bar{u}'_i \bar{u}_j})$, which has been neglected by many previous authors, is also important, although less so than the Leonard term. We have suggested a model of this term which appears to do an adequate job of approximating it.

8 Eddy viscosity models do only a fair job of matching the actual subgrid-scale Reynolds stresses, but they do a better job in matching the acceleration produced by the stresses and they are rather good at predicting the dissipation or energy transfer to the small scales.

9. All models of the eddy-viscosity class that we tested seem to be approximately equally good, and we are unable to choose among them on the basis of this study. The constant eddy-viscosity model is essentially what has been used by Orszag and co-workers, and our results partially explain their success. This point is probably closely related to the Reynolds-number independence of the large eddies.

10. Further improvements in subgrid-scale modeling are not likely to result from attempts to find improved formulas for the eddy viscosity. We have shown that the best any eddy-viscosity model could do is a relatively small improvement on the Smagorinsky model. Thus, turbulent kinetic energy and two-equation models which have been popular methods for time-average modeling are not promising approaches to subgrid-scale modeling.

11. We were unable to find improved models for the subgrid-scale Reynolds stresses, although a number of possibilities were tried. Further work in this direction could be fruitful.

Other recommendations that we would like to make are:

13. The effects of strain and shear on turbulence are very important, as they occur in essentially every flow of technological interest. The approach of this report ought to be extended to include those cases.

14. The subgrid-scale Reynolds stresses play a role in large-eddy simulations similar to that which the usual Reynolds stresses play in time- or ensemble-average calculations. We therefore suspect that analogous models ought to be equally valid in the two cases. Further work is needed to substantiate this suspicion, but, should it prove to be the case, our work would have important consequences for turbulence modeling in general.

15. One can derive exact equations for the subgrid scale Reynolds stresses. Using the approach of the present report, we can evaluate all of the terms in this equation and thus determine their importance and examine methods of modeling them.

16. The approach used in this report can be applied directly to the testing of time- and ensemble-average models. If we can compute flows in which the modeled effects are present, we could test the models in a manner similar to that used in the present work.

References

Batchelor, G. K., The Theory of Homogeneous Turbulence (Cambridge University Press, 1953), p. 118.

Cochran, W. T., et al , "What is the fast Fourier transform?" Proc. IEEE, Vol. 55, 10, p. 1664 (1967).

Comte-Bellot, G., and Corrsin, S., "Simple Eulerian time correlation of full- and narrow-band velocity signals in grid-generated isotropic turbulence," J. Fluid Mech., Vol 48, part 2, 273 (1971).

Deardorff, J. W., "A numerical study of three-dimensional turbulent flow at large Reynolds numbers," J. Fluid Mech., 42, 453-480 (1970).

Deardorff, J. W., "On the magnitude of the subgrid scale eddy coefficient," J. Comp. Phys , 7, 120 (1971).

Fox, D. G., and Lilly, K. L., "Numerical simulation of turbulent flows," Reviews of Geophysics and Space Physics, 10, No. 1 (1972).

Hirt, C. W., "Computer studies of time-dependent turbulent flows," Phys. Fluids, Suppl. 2, 219 (1969).

Kolmogoroff, A. N., "Dissipation of energy in locally isotropic turbulence," C. R. Acad. Sci. U.R.S.S., 32, 16 (1941)

Kwak, D., "Three-dimensional, time-dependent computation of turbulent flow," Report No. TF-5, Mechanical Engineering Department, Stanford University (1975).

Leonard, A., "On the energy cascade in large-eddy simulations of turbulent fluid flows," Report No. TF-1, Mechanical Engineering Department, Stanford University; or Advances in Geophysics, Vol. 18A, p 237 (1973).

Lilly, D. K., "On the application of the eddy viscosity concept in the inertial sub-range of turbulence," NCAR Manuscript No. 123 (1966).

Reynolds, W. C., "Computation of Turbulent Flows," Report No. TF-4, Mechanical Engineering Department, Stanford University (1975).

Richtmyer, R. D., and Morton, K. W., Difference Methods for Initial Value Problems (Interscience Publishers, 1967), 2nd ed.

Roache, P J., Computational Fluid Dynamics (Hermosa Publishers, 1972) p. 87.

Smagorinsky, J , "General circulation experiments with the primitive equations," Mo. Weather Rev , Vol. 93, 3, p. 99 (1963).

Appendix A

Higher-Order Time Methods

In Chapter 2 we developed a third-order time-advancement procedure. This method is a predictor-corrector method which requires only one evaluation of the time derivative per time step. In this appendix we look at two families of related methods in a more general way. Since the equations we deal with are parabolic with respect to the time variable, it is sufficient for initial study to consider only the ordinary differential equation

$$\dot{u} = \alpha u \tag{A.1}$$

where the dot denotes time differentiation and α may be complex.

The two most important questions relative to a numerical method are accuracy and stability. Accuracy is usually defined by assuming that if $u(0)$ were known exactly then the computed value of $u(\Delta)$, which we call $\hat{u}(\Delta)$, is related to the exact value by

$$\hat{u}(\Delta) \simeq u(\Delta) + \text{const. } \Delta^{n+1} u^{(n+1)} \tag{A.2}$$

where $u^{(n+1)}$ is the $(n+1)$st derivative of u. A method with this property is called n^{th} order. Loosely defined, stability means that the computation does not blow up. One common definition is that when the method it applied to Eq. (A.1) with α having negative real part, it does not produce a growing solution. For most methods, stability depends on the size of the step chosen, i.e., it is <u>conditionally stable</u>. Implicit methods may be unconditionally stable, but they are difficult to apply to nonlinear problems.

An approach to these questions is to look at the solutions of the difference equations. Although Eq. (A.1) has only one solution $e^{\alpha t}$, the difference equations may have multiple solutions. One of these approximates the solution of the differential equation with the desired

accuracy, the others are called parasitic solutions. All roots must have magnitude < 1 for stability.

For our purposes we want a method with the following properties

1. High accuracy -- this allows a large time step with acceptable error.
2. Stability -- the method must be stable for time steps as big as necessary for the accuracy requirement.
3. Few function evaluations -- in partial differential equation solving, the "functions" are partial derivatives and are costly to evaluate.
4. Minimum number of different values of variables required -- in partial differential equation solving, the "variables" are large arrays which require considerable memory See Chapter 2 on this point.

The popular Runge-Kutta method has the first two properties but not the last two. Essentially what we will do is accept poorer (but sufficient) stability in exchange for properties 3 and 4.

Two Evaluation Methods

The proposed methods are two-step (two previous values required) predictor-corrector type methods. The most general such method is

$$\begin{aligned} u_* &= \alpha_1 u_n + \alpha_2 \Delta\dot{u}_n + \alpha_3 u_{n-1} + \alpha_4 \Delta\dot{u}_{n-1} \quad , \\ u_{n+1} &= \beta_1 u_n + \beta_2 \Delta\dot{u}_n + \beta_3 u_{n-1} + \beta_4 \Delta\dot{u}_{n-1} + \beta_5 u_* + \beta_6 \Delta\dot{u}_* \quad . \end{aligned} \tag{A.3}$$

These can be combined to give

$$u_{n+1} = a_0 u_n + a_1 \Delta\dot{u}_n + a_2 \Delta^2 u_n + b_0 u_{n=1} + b_1 \Delta\dot{u}_{n=1} + b_2 \Delta^2 u_{n-1} \tag{A.4}$$

where

$$\begin{aligned} a_0 &= \beta_1 + \alpha_1\beta_5 & b_0 &= \beta_3 + \alpha_3\beta_5 \\ a_1 &= \beta_2 + \alpha_2\beta_5 + \alpha_1\beta_6 & b_1 &= \beta_4 + \beta_5\alpha_4 + \beta_6\alpha_3 \\ a_2 &= \alpha_2\beta_6 & b_2 &= \alpha_4\beta_6 \end{aligned} \tag{A.5}$$

Thus, although there are ten constants (α_1, β_1), only six combinations actually matter, four constants can be chosen arbitrarily to make the

method as simple as possible. By choosing the a_i, b_i properly, it is possible to obtain a fifth-order method, the method so obtained is highly unstable. So we will give up one order of accuracy to obtain stability. Applying the method (A.3) to Eq. (A.1) and looking for solutions of the type $u_n = \rho^n$, we find that ρ must be a root of the quadratic equation.

$$\rho^2 - (a_2(\alpha\Delta)^2 + a_1(\alpha\Delta) + a_0)\rho - (b_2(\alpha\Delta)^2 + b_1(\alpha\Delta) + b_0) = 0 \quad (A.6)$$

For the method to be stable at all, i.e., for both roots of this equation to be smaller than unity as $\Delta \to 0$, we must have

$$|b_0| \leq 1$$

For minimal accuracy, i.e., that one root approach unity as $\Delta \to 0$, we must have

$$a_0 = 1 - b_0$$

To obtain higher-order accuracy, we match the coefficients of the Taylor series of one of the roots to that of $e^{\alpha\Delta}$ and find:

1st order	$-a_1 + b_0 - b_1 = -1$
2nd order	$-2a_2 = 2a_1 + b0 - 2b_2 = -3$
3rd order	$6a_2 + 3a_1 - b_0 = 7$
4th order	$12a_2 + 4a_1 - b_0 = 15$

Solving, we have

$$a_1 = \frac{b_0 - 1}{2}, \quad a_2 = \frac{17 - b_0}{12},$$

$$b_1 = \frac{b_0 + 3}{2}, \quad b_2 = \frac{b_0 + 7}{12}.$$

By choosing values of b_0 within the allowed range, we can generate a family of methods. It turns out that $b_0 = 1$ has the poorest stability properties, while $b_0 = -1$ has the least accuracy

The stability bounds for selected values of b_0 are given in Table A.1 They were computed in the manner described in Chapter 2. We see that maximum stability is obtained at $b_0 \simeq .25$, and the allowable time step is considerably below that of the fourth-order Runge Kutta method, for which $|\alpha\Delta|_{max} = 3\ 8$. However, in our calculations the effective value of $|\alpha\Delta|$ is approximately 0.2-0.3 for accuracy reasons, so this stability limit causes no problem.

Table A.1

Stability Bounds

b_0	$\lvert\alpha\Delta\rvert_{max}$
-1.	0
0.	51
0.2	.60
0.3	.60
0.5	.55
1.0	0.

We also solved Eq (A.1) using this method with $b_o = 0$, with two different sets of constants. Within roundoff error, both methods produced identical results. It should be noted that, to minimize roundoff errors, one should choose sets of (α_1,β_1) with the smallest values possible.

One-Evaluation Methods

The methods described above require two evaluations of derivatives per step; i.e., $\dot{u}_*$ and $\dot{u}_n$ both need to be computed. To avoid this we would need either $\beta_6 = 0$ or $\alpha_2 = \alpha_4 = \beta_2 = \beta_4 = 0$, either of which is incompatible with Eqs. (A.5). In order to obtain a single-evaluation method, we therefore use

$$u_{*n+1} = \alpha_1 u_n + \alpha_2 \Delta\dot{u}_{*n} + \alpha_3 u_{n-1} + \alpha_4 \Delta\dot{u}_{*n-1}$$

$$u_{n+1} = \beta_1 u_n + \beta_2 \Delta\dot{u}_{*n} + \beta_3 u_{n-1} + \beta_4 \Delta\dot{u}_{*n-1} + \beta_5 u_{*n+1} + \beta_6 \Delta\dot{u}_{*n+1} \qquad (A\ 7)$$

Which requires the evaluation of $\dot{u}_*$ only. We now assume solutions of the form

$$\begin{pmatrix} u_{*n} \\ u_n \end{pmatrix} = \begin{pmatrix} u_* \\ u \end{pmatrix} \rho^n$$

and find that ρ satisfies a quartic equation

$$\rho^4 - \Big(\eta_1 + \eta_2(\alpha\Delta)\Big)\rho^3 - \Big(\eta_3 + \eta_4(\alpha\Delta)\Big)\rho^2 + \eta_5(\alpha\Delta)\rho + \eta_6(\alpha\Delta) = 0 \tag{A.8}$$

where

$$\begin{aligned} \eta_1 &= \beta_1 + \beta_5\alpha_1 \ , & \eta_4 &= -\alpha_2\beta_1 + \alpha_4 + \alpha_1\beta_2 + \alpha_3\beta_6 \ , \\ \eta_2 &= \alpha_2 + \alpha_1\beta_6 \ , & \eta_4 &= \alpha_2\beta_3 + \alpha_4\beta_1 - \alpha_1\beta_4 - \alpha_3\beta_2 \ , \\ \eta_3 &= \beta_3 + \alpha_3\beta_5 \ , & \eta_6 &= \alpha_4\beta_3 - \alpha_3\beta_4 \ . \end{aligned} \tag{A.9}$$

So, again, fifth-order is the maximum possible. The method so obtained is again unstable, so we must settle for fourth-order. We now find that for stability

$$0 \leq \eta_1 \leq 2 \ ,$$

and accuracy requires that the other parameters be related to η_1 by.

$$\begin{aligned} \eta_2 &= \frac{8}{3} - \frac{3}{8}\eta_1 \ , & \eta_4, &= -\frac{4}{3} - \frac{5}{24}\eta_1 \ , \\ \eta_3 &= 1 - \eta_1 \ , & \eta_6 &= \frac{1}{3} + \frac{1}{24}\eta_1 \ . \\ \eta_4 &= -\frac{5}{3} - \frac{19}{24}\eta_1 \ , \end{aligned}$$

The stability limits for these methods is shown in Table A.2. Maximum stability is obtained with $\eta_1 = 2$, and the allowable time step is only slightly smaller than that for the two-evaluation method

In testing these methods, however, we found that these methods do not always produce the accuracy that one might expect. In particular, if an arbitrary version of this method is used rather poor results are obtained unless the method is started carefully. The problem can be cured

Table A.2

Stability Limits

η_1	$\lvert\alpha\Delta\rvert_{max}$
0.	0.
1.	0.30
1.5	0.42
2.0	0.53

by requiring that the predictor step be accurate. The predictor can be made third-order accurate, but only if $\eta_1 = 0$, which results in instability. We therefore recommend that the method be used with second-order predictors. For these the method is uniquely defined by the predictor step. Two possibilities are (1) using the leap-frog method as a predictor ($\eta_1 = 16/9$).

$$u_{*n+1} = u_{n-1} + 2\Delta\dot{u}_{n+1}$$
$$u_{n+1} = \frac{1}{9}(16u_n - 7u_{*n+1}) + \frac{\Delta}{27}(46\dot{u}_{*n} - 11\dot{u}_{*n-1} + 13\dot{u}_{*n+1}) \tag{A.10}$$

and (2) using Adams Bashforth as the predictor ($\eta_1 = 20/11$):

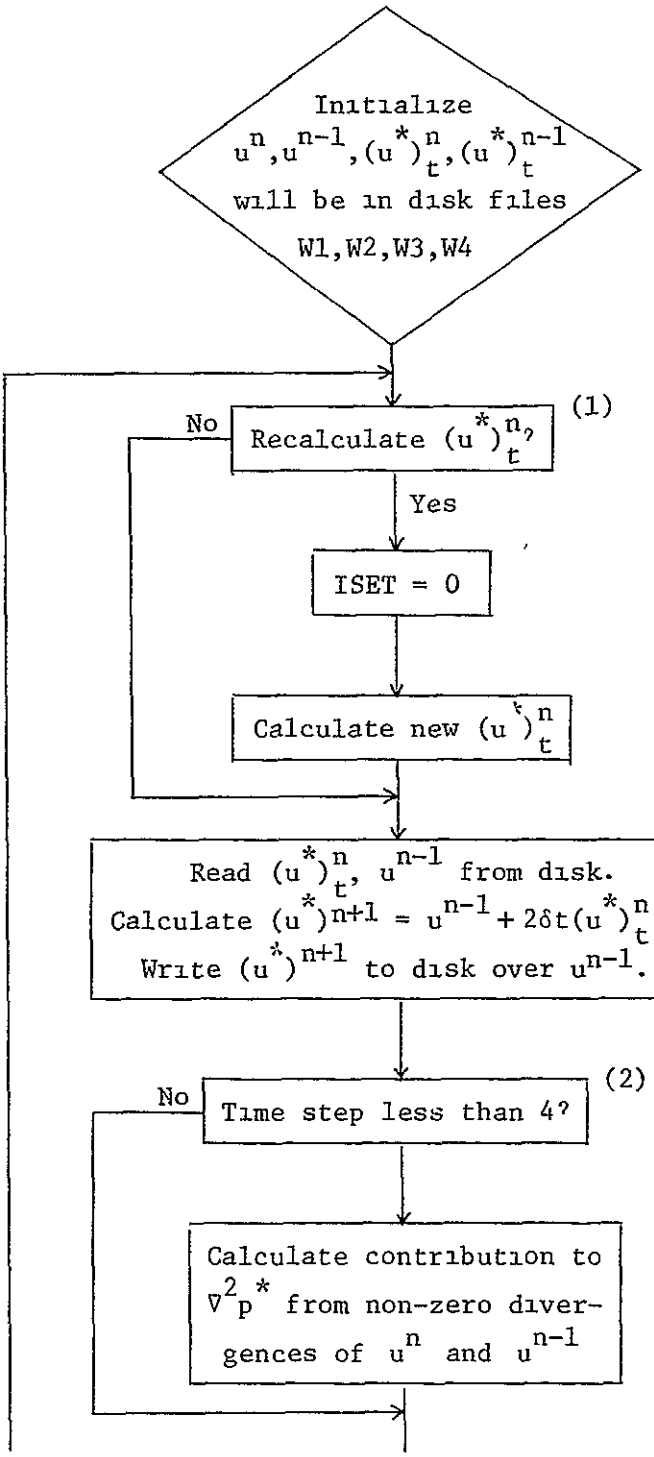
$$u_{*n+1} = u_n + \frac{\Delta}{2}(3\dot{u}_{*n} - \dot{u}_{*n-1})$$
$$u_{n+1} = \frac{1}{11}(-9u_{*n-1} + 20u_{*n+1}) + \frac{\Delta}{33}(-86\dot{u}_{*n} + 16\dot{u}_{*n-1} + 16\dot{u}_{*n+1}) \tag{A.11}$$

Good results were obtained with both of these methods. However, we note that both methods contain some large coefficients in the corrector steps which is undesirable from the point of view of roundoff error propagation, and we have used the safer third-order method described in Chapter 2 in our calculation.

Appendix B

PROGRAMS AND FLOW CHART

Flow Chart of Main Program

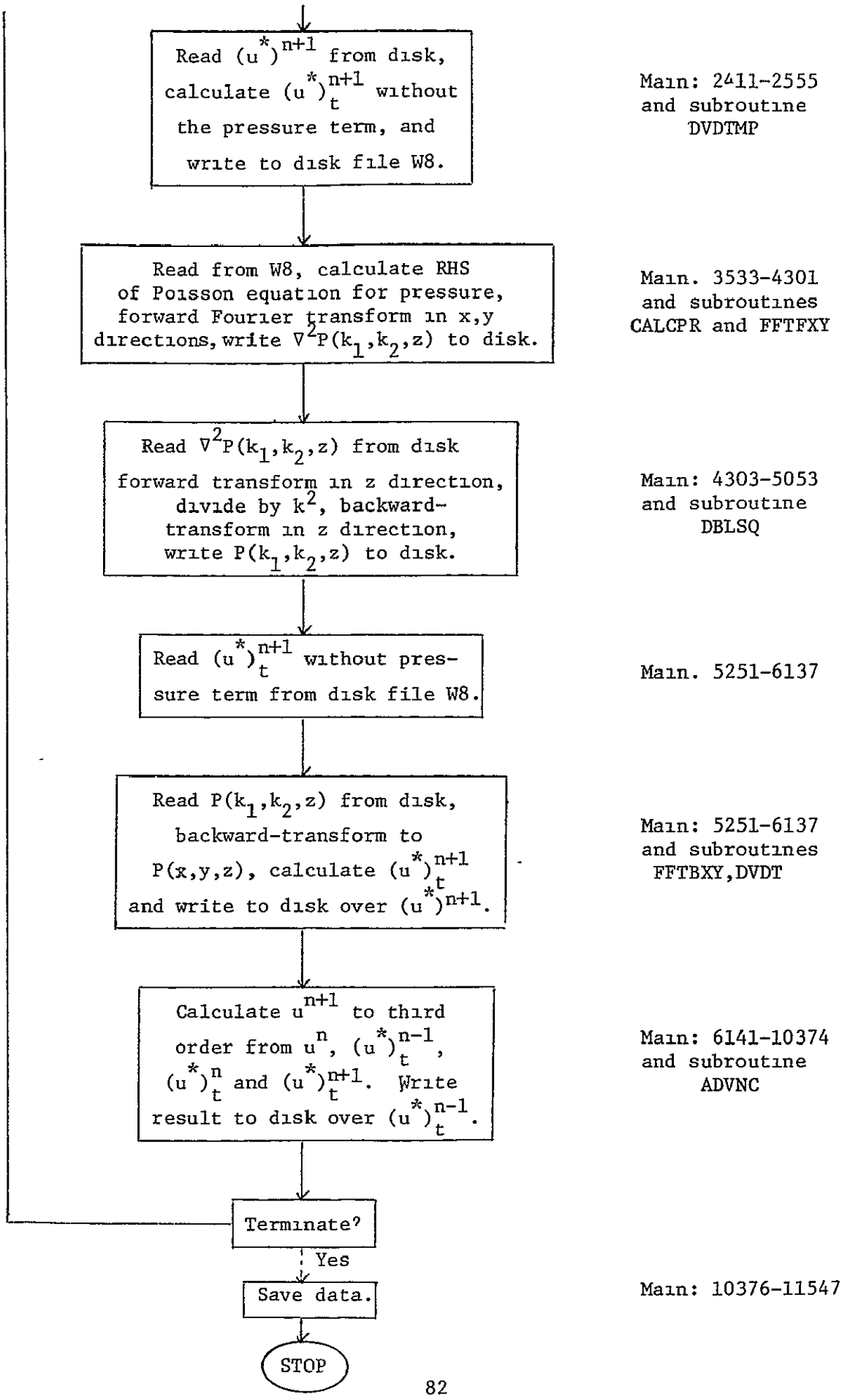


Location in Listing

Main. 1 → 1001

Main 1166-1541

Main: 1543-2407


Read $(u^*)^{n+1}$ from disk, calculate $(u^*)_t^{n+1}$ without the pressure term, and write to disk file W8.
Main: 2411-2555 and subroutine DVDTMP
Read from W8, calculate RHS of Poisson equation for pressure, forward Fourier transform in x,y directions, write $\nabla^2 P(k_1,k_2,z)$ to disk.
Main. 3533-4301 and subroutines CALCPR and FFTFXY
Read $\nabla^2 P(k_1,k_2,z)$ from disk forward transform in z direction, divide by k^2, backward-transform in z direction, write $P(k_1,k_2,z)$ to disk.
Main: 4303-5053 and subroutine DBLSQ
Read $(u^*)_t^{n+1}$ without pressure term from disk file W8.
Main. 5251-6137
Read $P(k_1,k_2,z)$ from disk, backward-transform to $P(x,y,z)$, calculate $(u^*)_t^{n+1}$ and write to disk over $(u^*)^{n+1}$.
Main: 5251-6137 and subroutines FFTBXY,DVDT
Calculate u^{n+1} to third order from u^n, $(u^*)_t^{n-1}$, $(u^*)_t^n$ and $(u^*)_t^{n+1}$. Write result to disk over $(u^*)_t^{n-1}$.
Main: 6141-10374 and subroutine ADVNC
Terminate?
Yes
Save data.
Main: 10376-11547
STOP

(1) For stability reasons we sometimes recalculate $(u^*)_t^n$ using the corrected third-order u^n.

(2) For the first three time steps only, the divergences of the previous time steps are not necessarily zero.

System Subroutines Used in Main and Start

1. MEMREQ

   Call MEMREQ(NWORDS) request NWORDS of LCM space for this program.

2. FORQTS

   Call FORQTS(W1,RQ1) reserves space at RQ1 for references to disk file W1.

3 CREATE

   Call CREATE(W1,,,,,,,,NSECT) creates the disk file W1 with length NSECT sectors where a sector is 512 words.

4. IDONE

   The statement I=IDONE(W2) sets I = 0 if there is outstanding I/O to be completed to or from file W1, otherwise I = 1.

5. IRANR

   The statement I=IRANR(RQ1,A(N),NWORDS,NSECT,W1) causes NWORDS to be transferred from disk file W2 starting with sector NSECT (Sector number 0 is the start of the file; there are 512 words per sector), to LCM starting at address A(N).

6. IRANW

   I=IRANW is analogous write statement to IRANR.

7. FFT2

   Call FFT2(A(I1),B(Z2),N, INC) performs a fast-Fourier transform of length N on the real data starting at location A(I1) and the imaginary data starting at location B(I2) The data are incremented by INC, -INC implies a forward transform and +INC implies a backward transform.

The Main Program

The main program assumes that disk files containing the first two time steps, i.e., u^{n-1}, $(u^*)_t^{n-1}$, u^n, and $(u^*)_t^n$, exist

```
      PROGRAM GAP(OUT,FSET5,FSET6,FSET7,FSET8,FSET9)
2     IMPLICIT INTEGER (Z)
2     LCM/BB1/DUM1(98304)
2     LCM/BB2/DUM2(98304)
2     LCM/BB3/DUM3(65536)
C     MAX LCM BLOCK=126976
2     DIMENSION VLG(4096,3,8),DLG(64,64,4),PLG(4096,2,4),PZ1(2048,2,24),
     2PZ2(2048,2,24),PZ3(2048,2,16),PHAT(4096,2,6),PC(64,64,6),WLG(4096,
     33,8),DULG(4096,3,2),          PHATA(4096,2,2),DVLG(4096,3,8)
2     DIMENSION UTNM1(24576),UTN(24576),UTNP1(24576),UN(24576),
     2VTNM1(8192),VTN(8192),VTNP1(8192),VN(8192)
2     DIMENSION WTNM1(24576),WTN(24576),WTNP1(24576),WN(24576)
2     DIMENSION USM(64,20,5),VSM(64,20,5),WSM(64,20,5),          P(64,6
     24,2),PZ(256,64,2),PA(4096,2 ),PR(64,64),  DU(64,16),  DV(64,16),
     3DW(64,16),U(64,20,5),V(64,20,5),W(64,20,5),PD(64,20,5)
2     DIMENSION RQ1(20),RQ2(20),RQ3(20),RQ4(20),RQ5(20),RQ6(20),RQ7(20)
     2,RQ8(20),SPR(64),SPI(64)
2     DIMENSION       S(255),C(255),S64(255),C64(255)
2     DIMENSION ITAPE(2),I05TAPE(5),I06TAPE(5),I1TAPE(5),I2TAPE(5),
     2I3TAPE(5),I4TAPE(5),I5TAPE(5),I6TAPE(5),I7TAPE(5),I8TAPE(5),
     3I07TAPE(5),I08TAPE(5)
C      DISK FILES
C     W1 CONTAINS 64 PLANES OF V,V,W. EACH PLANE CONSISTS OF 24 SECTIONS=12288
C     WORDS.  IE 1536 SECTIONS.  PLANE 63 STARTS AT NSECT=0, PLANE I STARTS AT
C     24*(I+1).  IF((24*I+1).GT.1512)NSECT=24$(I+1)-1536
C     W2 IS IDENTICAL TO W1
C     D1 CONSSITS OF 64 PLANES OF DIV, EACH PLANE CONTAINS 8 SECTIONS=4096 WORDS
C     IE 512 SECTIONS.  PLANE 1 STARTS AT NSECT=0, PLANE I STARTS AT
C     NSECT=8*(I-1)
C     PUPR CONTAINE 64 PLANES OF UPPER HALF OF J, EACH PLANES CONSISTS OF 8
C     SECTIONS=4096 WORDS.  PLANE 63 STARTS AT NSECT=0, PLANE I STARTS AT NSECT
C     =8*(I+1), IF(I+1)*8.GT.504)NSECT=NSECT-512
C     PLWR IS SAME AS PUPR
2     COMMON DUMSM(32768)
2     EQUIVALENCE(VLG(1),DUM1(1)),(DVLG(1),DUM2(1)),(PLG(1),DUM3(1)),
     2(DLG(1),DUM3(32769)),(WLG(1),DUM2(1))
2     EQUIVALENCE (PZ1(1),DUM1(1)),(PZ2(1),DUM2(1)),(PZ3(1),DUM3(1))
2     EQUIVALENCE (DULG(1),DUM1(73729)),(PHAT(1),DUM2(1)),(PHATA(1),DUM2
     2(32769)),(PC(1),DUM2(49153))
2     EQUIVALENCE(UTNM1(1),DUM1(1)), (UTN(1),DUM1(24577)),(UTNP1(1),
     2DUM1(49153)),(UN(1),DUM1(73729)),(VN(1),DUMSM(1)),(VTNM1(1),DUMSM(
     34097)), (VTN(1),DUMSM(8193)),(VTNP1(1),DUMSM(12289))
2     EQUIVALENCE(WTNM1(1),DUM2(1)), (WTN(1),DUM2(24577)),(WTNP1(1),
     2DUM2(49153)),(WN(1),DUM2(73729))
2     EQUIVALENCE          (DUMSM(1),SPI(1)),                    (P(1
     2  ),DUMSM(I    )),(PZ(1,1,1  ),DUMSM(1)),(PA(1,1  ),DUMSM(4097)),(
     3PR(1,1),DUMSM(1)),(USM(1),DUMSM(4097 )),(VSM(1),DUMSM(10497)),
     4(WSM(1)     ,DUMSM(16897)),(  DU(1,1),DUMSM(    1)),(  DV(1,1),DUMSM
     5(1025)),(  DW(1,1),DUMSM( 2049)),(U(1,1,1),DUMSM( 3073)),(V(1,1,1)
     6,DUMSM(9473 )),(W(1,1,1),DUMSM(15873)),(PD(1,1,1),DUMSM(22273))
      DATA RQ1/20*0./,RQ2/20*0./,RQ3/20*0./,RQ4/20*0./,RQ5/20*0./
      DATA RQ6/20*0./,RQ7/20*0./,RQ8/20*0./
      DATA W1,W2,D1,PUPR,PLWR/4LW1,2LW2,2LD1,4LPUPR,4LPLWR/
      DATA W3,W4,W8/2LW3,2LW4,2LW8/
      DATA ITAPE/1,4LTAPE/,I05TAPE/4,3*0,8LXX014477/,I06TAPE/4,3*0,
```

```
      28LXX014883/,I1TAPE/4,3*0,8LXX005112/,I2TAPE/4,3*0,8LXX005769/
       DATA I3TAPE/4,3*0,8LXX005663/,I4TAPE/4,3*0,8LXX005322/,I5TAPE/4,
      2 3*0,8LXX007039/,I6TAPE/4,3*0,8LXX009091/,I7TAPE/5*0/
      3,I8TAPE/5*0/
       DATA I07TAPE/5*0/,I08TAPE/5*0/
    2  CALL MEMREQ(262144,1)
    4  CALL FORQTS(W1,RQ1)
    6  CALL FORQTS(W2,RQ2)
   10  CALL FORQTS(W3,RQ3)
   12  CALL FORQTS(W4,RQ4)
   14  CALL FORQTS(W8,RQ8)
   16  CALL FORQTS(D1,RQ5)
   20  CALL FORQTS(PUPR,RQ6)
   22  CALL FORQTS(PLWR,RQ7)
   24  DO 3 I=1,8
   31 3 DUM1(I)=0.
   34  CALL CREATE(W1  ,U,RT,0,0,0,0,0,0,1536)
   46  I=IRANW(RQ1,DUM1(1),1,1536,W1)
   61  CALL CREATE(W2  ,U,RT,0,0,0,0,0,0,1536)
   73  I=IRANW(RQ2,DUM1(2),1,1536,W2)
  106  CALL CREATE(W3  ,U,RT,0,0,0,0,0,0,1536)
  120  I=IRANW(RQ3,DUM1(3),1,1536,W3)
  133  CALL CREATE(W4  ,U,RT,0,0,0,0,0,0,1536)
  145  I=IRANW(RQ4,DUM1(4),1,1536,W4)
  160  CALL CREATE(W8  ,U,RT,0,0,0,0,0,0,1536)
  172  I=IRANW(RQ8,DUM1(8),1,1535,W8)
  205  CALL CREATE(D1  ,U,RT,0,0,0,0,0,0,512)
  217  I=IRANW(RQ5,DUM1(5),1,511,D1)
  232  CALL CREATE(PUPR,U,RT,0,0,0,0,0,0,512)
  244  I=IRANW(RQ6,DUM1(6),1,511,PUPR)
  257  CALL CREATE(PLWR,U,RT,0,0,0,0,0,0,512)
  271  I=IRANW(RQ7,DUM1(7),1,511,PLWR)
  304  912 I=IDONE(W1)*IDONE(W2)*IDONE(W3)*IDONE(W4)*IDONE(W8)*IDONE(D1)
      2*IDONE(PUPR)*IDONE(PLWR)
  332      IF(I.NE.8)GO TO 912
  334      CALL OPEN(5LFSET5,0,2340008)
  337      DO 920 K=1,8
  341      NSECT=(K-1)*192
  343      M=MOD(K,2)
  347      DO 910 J=1,3
  351      JK=(J-1)*32768+1
  354      READ(5)(DUMSM(I),I=1,32768)
  361      IF(M.EQ.0)GO TO 909
  362      SMALL OUT(DUMSM(1),DUM1(JK),32768)
  371      GO TO 910
  371  909 SMALL OUT(DUMSM(1),DUM2(JK),32768)
  400  910 CONTINUE
  402  911 I=IDONE(W1)
  404      IF(I.NE.1)GO TO 911
  406      IF(M.EQ.0)GO TO 919
  407      I=IRANW(RQ1,DUM1(1),98304,NSECT,W1)
  422      GO TO 920
  422  919 I=IRANW(RQ1,DUM2(1),98304,NSECT,W1)
  436  920 CONTINUE
  440      IFIRST=0
```

```
441         IF(IFIRST.EQ.1)GO TO 941
443         DO 940 K=1,8
444         NSECT=(K-1)*192
446         M=MOD(K,2)
452         DO 93 J=1,3
454         JK=(J-1)*32768+1
457         READ(5)(DUMSM(I),I=1,32768)
464         IF(M.EQ.0)GO TO 929
465         SMALL OUT(DUMSM(1),DUM1(JK),32768)
474         GO TO 930
474     929 SMALL OUT(DUMSM(1),DUM2(JK),32768)
503     930 CONTINUE
505     931 I=IDONE(W2)+IDONE(W1)
512         IF(I.NE.2)GO TO 931
514         IF(M.EQ.0)GO TO 939
515         I=IRANW(RQ2,DUM1(1),98304,NSECT,W2)
530         GO TO 940
530     939 I=IRANW(RQ2,DUM2(1),98304,NSECT,W2)
544     940 CONTINUE
546     941 CALL AFSREL(5LFSET5)
550         CALL OPEN(5LFSET6,0,234000B)
553         DO 960 K=1,8
555         NSECT=(K-1)*192
557         M=MOD(K,2)
563         DO 95 J=1,3
565         JK=(J-1)*32768+1
570         READ(6)(DUMSM(I),I=1,32768)
575         IF(M.EQ.0)GO TO 949
576         SMALL OUT(DUMSM(1),DUM1(JK),32768)
605         GO TO 950
605     949 SMALL OUT(DUMSM(1),DUM2(JK),32768)
614     950 CONTINUE
616     951 I=IDONE(W3)+IDONE(W2)
623         IF(I.NE.2)GO TO 951
625         IF(M.EQ.0)GO TO 959
626         I=IRANW(RQ3,DUM1(1),98304,NSECT,W3)
641         GO TO 960
641     959 I=IRANW(RQ3,DUM2(1),98304,NSECT,W3)
655     960 CONTINUE
657         IF(IFIRST.EQ.1)GO TO 981
661         DO 980 K=1,8
663         NSECT=(K-1)*192
665         M=MOD(K,2)
671         DO 97 J=1,3
673         JK=(J-1)*32768+1
676         READ(6)(DUMSM(I),I=1,32768)
703         IF(M.EQ.0)GO TO 969
704         SMALL OUT(DUMSM(1),DUM1(JK),32768)
713         GO TO 970
713     969 SMALL OUT(DUMSM(1),DUM2(JK),32768)
722     970 CONTINUE
724     971 I=IDONE(W4)+IDONE(W3)
731         IF(I.NE.2)GO TO 971
733         IF(M.EQ.0)GO TO 979
734         I=IRANW(RQ4,DUM1(1),98304,NSECT,W4)
```

```
 747        GO TO 980
 747    979 I=IRANW(RQ4,DUM2(1),98304,NSECT,W4)
 763    980 CONTINUE
 765    981 CALL AFSREL(5LFSET6)
 767        DO 1 I=1,32768
 774      1 DUMSM(I)=0.
 776        W=3.1415926535898/2048.
 777        DO 4 I=1,255
1001        S(I)=SIN((I-1)*W)
1010      4 C(I)=COS ((I-1)*W)
1021        W=3.1415926535898/32.
1022        DO 5 I=1,255
1024        S64(I)=SIN((I-1)*W)
1033      5 C64(I)=COS((I-1)*W)
1045        DELT=.0073
1046        OTDT=DELT/12.
1047        TTDT=2.*DELT/3.
1051        DELZ=20./64.
1053        C1=1./(12.*DELZ)
1055        X=20.
1057        C6=7456540.442/X**2
1061        C12=C1
1062        CN=.1453
1064        C7=12./(5.*DELT)
1066        C8=1./(12.*DELZ)
1071        C11=1./(12.*DELZ**2)
1073        TDT=4.*DELT
1074        ITIME=40
1075        ISET=
1077   1001 CONTINUE
1077        IM=MOD(ITIME,4)+1
1104        IF(ITIME.LT.2)GO TO 400
1105        IF(MOD(ITIME,8).GT.3)ISET=1
1113        IF(ISET.EQ.0)GO TO 400
1114        DO 873 I=1,64
1121        SPR(I)=0.
1122    873 SPI(I)=0.
1123        RMSD=.
1123        RMSU=0.
1124        RMSV=0.
1125        RMSW=0.
1125        SK1=.
1126        SK2=0.
1126        UMAX=0.
1127        VMAX=0.
1127        WMAX=0.
1130        CRMSU=0.
1130        CRMSV=0.
1131        CRMSW=0.
1131        CUMAX=0.
1132        CVMAX=0.
1132        CWMAX=0.
      C
      C     ************       PHASE I     ***********
      C
```

```
      C     START WITH PLANES 63,64,1,2,3,4,5,6 IN LCM
      C     V(1) AT SEGMENT 0, V(2) AT SEGMENT 24, V(N) AT SEGMENT 24*N-IJ
      C     V(64) AT SEGMENT 1512,  EACH V CONTAINS 12288 WORDS
      C     VARIABLE LOCATIONS AS FOLLOWS AT START OF TIME STEP
      C     TIME=  1          2          3          4
      C     W1  U(N-1)     UT(N)      UT(N-1)    U(N)
      C     W2  UT(N-1)    U(N)       U(N-1)     UT(N)
      C     W3  U(N)       U(N-1)     UT(N)      UT(N-1)
      C     W4  UT(N)      UT(N-1)    U(N)       U(N-1)
1133    400 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)+IDONE(W8)+IDONE(D1)
           2+IDONE(PUPR)*IDONE(PLWR)
1161        IF(I.NE.8)GO TO 400
1163        IF(ITIME.LT.0)GO TO 1901
1165        IF(ISET.EQ.0)GO TO 1902
1166        DO 155 N=1,8
1167        NSECT=(N-1)*192
1172    130 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)
1205        IF(I.NE.4)GO TO 130
1207        GO TO (131,132,133,134)IM
1217    131 I=IRANR(RQ1,VLG(1),98304,NSECT,W1)
1232        I=IRANR(RQ4,DVLG(1),98304,NSECT,W4)
1246        GO TO 139
1246    132 I=IRANR(RQ3,VLG(1),98304,NSECT,W3)
1261        I=IRANR(RQ1,DVLG(1),98304,NSECT,W1)
1275        GO TO 139
1275    133 I=IRANR(RQ2,VLG(1),98304,NSECT,W2)
1310        I=IRANR(RQ3,DVLG(1),98304,NSECT,W3)
1324        GO TO 139
1324    134 I=IRANR(RQ4,VLG(1),98304,NSECT,W4)
1337        I=IRANR(RQ2,DVLG(1),98304,NSECT,W2)
1353    139 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)
1366        IF(I.NE.4)GO TO 139
1370        DO 141 K=1,8
1372        SMALL IN(DUMSM(),VLG(1,1,K),12288)
1401        SMALL IN(DUMSM(12289),DVLG( ,1,K),12288)
1410        DO 140 I=1,12288
1415    140 DUMSM(I)=DUMSM(I)+TDT*DUMSM(I+12288)
1420    141 SMALL OUT(DUMSM(1),VLG(1,1,K),12288)
1430        GO TO(151,152,153,154)IM
1440    151 I=IRANW(RQ1,VLG(1),98304,NSECT,W1)
1454        GO TO 155
1454    152 I=IRANW(RQ3,VLG(1),98304,NSECT,W3)
1470        GO TO 155
1470    153 I=IRANW(RQ2,VLG(1),98304,NSECT,W2)
1504        GO TO 155
1504    154 I=IRANW(RQ4,VLG(1),98304,NSECT,W4)
1520    155 CONTINUE
1522    156 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)
1535        IF(I.NE.4)GO TO 156
1537    157 I=IDONE(D1)
1541        IF(I.NE.1)GO TO 157
1543        IF(ITIME.GT.3)GO TO 600
1547        ID=1
1547        IMP=IM
1551        GO TO 499
```

```
1552   1901 IMP=4
1553        ID=1
1554        GO TO 499
1555   1902 IF(ITIME.GT.3)GO TO 600
1561        IF(IM.EQ.1)IMP=4
1564        IF(IM.EQ.2)IMP=1
1567        IF(IM.EQ.3)IMP=2
1572        IF(IM.EQ.4)IMP=3
1575        ID=1
1576    499 CONTINUE
1576        IF(IU.NE.1)I=IRANR(RQ5,DUM3(1),32768,0,D1)
1613        GO TO(461,462,463,464)IMP
1623    461 I=IRANR(RQ3,DUM1(1),98304,0,W3)
1637        GO TO 465
1637    462 I=IRANR(RQ2,DUM1(1),98304,0,W2)
1653        GO TO 465
1653    463 I=IRANR(RQ4,DUM1(1),98304,0,W4)
1667        GO TO 465
1667    464 I=IRANR(RQ1,DUM1(1),98304,0,W1)
1703    465 I=IDONE(D1)+IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)
1721        IF(I.NE.5)GO TO 465
1723        DO 490 M=1,16
1724        IF(MOD(M,2).EQ.0)GO TO 480
1730        NSECT=M*96
1731        NWORDS=98304
1733        J=1
1734        IF(M.EQ.15)NWORDS=49152
1737    475 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)
1752        IF(I.NE.4)GO TO 475
1754        GO TO (491,492,493,494)IMP
1764    491 I=IRANR(RQ3,DUM2(J),NWORDS,NSECT,W3)
2000        GO TO 473
2000    492 I=IRANR(RQ2,DUM2(J),NWORDS,NSECT,W2)
2014        GO TO 473
2014    493 I=IRANR(RQ4,DUM2(J),NWORDS,NSECT,W4)
2030        GO TO 473
2030    494 I=IRANR(RQ1,DUM2(J),NWORDS,NSECT,W1)
2044    473 IF((M.NE.15).OR.(NSECT.EQ.0))GO TO 479
2053        NSECT=0
2053        J=49153
2054        GO TO 475
2055    479 I=IDONE(D1)
2057        IF(I.NE.1)GO TO 479
2061        IF(M.EQ.1)GO TO 477
2063        J=M+1
2063        NSECT=(J/2-2)*64
2067        IOA=1
2070        IF(MOD(J,4).NE.0)IOA=32769
2075        I=IRANW(RQ5,DUM3(IOA),32768,NSECT,D1)
2111        GO TO 477
2111    480 NSECT=M*96
2113        NWORDS=98304
2115    476 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)
2130        IF(I.NE.4)GO TO 476
2132        IF(M.EQ.16)GO TO 477
```

```
2134        GO TO(495,496,497,498)IMP
2144    495 I=IRANR(RQ3,DUM1(1),NWORDS,NSECT,W3)
2160        GO TO 478
2160    496 I=IRANR(RQ2,DUM1(1),NWORDS,NSECT,W2)
2174        GO TO 478
2174    497 I=IRANR(RQ4,DUM1(1),NWORDS,NSECT,W4)
2210        GO TO 478
2210    498 I=IRANR(RQ1,DUM1(1),NWORDS,NSECT,W1)
2224    478 I=IDONE(D1)
2226        IF(I.NE.1)GO TO 478
2230        IF(IU.EQ.1)GO TO 477
2232        IF(M.EQ.16)GO TO 477
2234        IOA=1
2234        IF(MOD(M,4).NE.0)IOA=32769
2242        NSECT=(M/2)*64
2245        I=IRANR(RQ5,DUM3(IOA),32768,NSECT,D1)
2261    477 CONTINUE
2261        CALL DVRGNC(C12,RMSD,RMSU,RMSV,RMSW,SK1,SK2,UMAX,VMAX,WMAX,M,ID,DE
           >LT)
2277    490 CONTINUE
2301    484 I=IDONE(D1)
2303        IF(I.NE.1)GO TO 484
2305        I=IRANW(RQ5,DUM3(32769),32768,448,D1)
2320    489 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)+IDONE(D1)
2336        IF(I.NE.5)GO TO 489
2340        IF(ITIME.LT.0)GO TO 601
2341        IF(ISET.EQ.0)GO TO 608
2342        GO TO(501,502,503)ID
2351    501 ID=2
2352        IF(IM.EQ.1)IMP=2
2355        IF(IM.EQ.2)IMP=3
2360        IF(IM.EQ.3)IMP=4
2363        IF(IM.EQ.4)IMP=1
2366        GO TO 499
2367    502 ID=3
2370        IF(IM.EQ.1)IMP=3
2373        IF(IM.EQ.2)IMP=4
2376        IF(IM.EQ.3)IMP=1
2401        IF(IM.EQ.4)IMP=2
2404        GO TO 499
2405    503 CONTINUE
2405    100 CONTINUE
2405    460 I=IDONE(D1)
2407        IF(I.NE.1)GO TO 460
2411    600 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)+IDONE(W8)+IDONE(D1)
           >+IDONE(PUPR)+IDONE(PLWR)
2437        IF(I.NE.8)GO TO 600
2441        IF(ISET.EQ.0)GO TO 608
2442        GO TO(601,602,603,604)IM
2452    608 GO TO (602,603,604,601)IM
2462    601 I=IRANR(RQ1,VLG(1),98304,0,W1)
2476        GO TO 605
2476    602 I=IRANR(RQ3,VLG(1),98304,0,W3)
2512        GO TO 605
2512    603 I=IRANR(RQ2,VLG(1),98304,0,W2)
```

```
2526         GO TO 605
2526     604 I=IRANR(RQ4,VLG(1),98304,0,W4)
2542     605 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)
2555         IF(I.NE.4)GO TO 605
2557         ZM2=1
2560         IOA=1
2561         DO 620 IPLANE=1,64
     C       TRANSFER ROWS 45 TO 64 TO CALCULATE 47 TO 62
2562         N2=2817
2563         N3=1280
2563         N4=1
2564         NEXT=1
2565         M=1
2566     613 K=ZM2-1
2570         DO 621 Z=1,5
2572         K=K+1
2573         IF(K.GT.8)K=1
2576         SMALL IN(  U(1,N4,Z),VLG(N2,1,K),N3)
2612         SMALL IN(  V(1,N4,Z),VLG(N2,2,K),N3)
2626     621 SMALL IN(  W(1,N4,Z),VLG(N2,3,K),N3)
2644         GO TO(622,622,624,622)NEXT
2654     622 CALL DVDTMP(U,V,W,DU,DV,DW,M,CN,C8,C11,C12,ITIME,RMSD,RMSU,RMSV,
            2RMSW,SK1,SK2,UMAX,VMAX,WMAX,SPR,ISET)
2704         SMALL OUT(DU(1),DVLG(2945,1,IOA),1024)
2714         SMALL OUT(DV(1),DVLG(2945,2,IOA),1024)
2723         SMALL OUT(DW(1),DVLG(2945,3,IOA),1024)
2732         DO 614 J=1,4
2733         JJ=J+16
2734         DO 614 I=1,64
2736         DO 614 Z=1,5
2752         U(I,J,Z)=U(I,JJ,Z)
2754         V(I,J,Z)=V(I,JJ,Z)
2755     614 W(I,J,Z)=W(I,JJ,Z)
     C       CALCULATE ROWS 63,64 AND 1 TO 14
2763         N2=1
2763         N3=1024
2764         N4=5
2765         NEXT=2
2766         M=2
2767         GO TO 613
2770     623 CALL DVDTMP(U,V,W,DU,DV,DW,M,CN,C8,C11,C12,ITIME,RMSD,RMSU,RMSV,
            2RMSW,SK1,SK2,UMAX,VMAX,WMAX,SPR,ISET)
3020         SMALL OUT(DU(1),DVLG(3969,1,IOA),128)
3030         SMALL OUT(DV(1),DVLG(3969,2,IOA),128)
3037         SMALL OUT(DW(1),DVLG(3969,3,IOA),128)
3046         SMALL OUT(DU(129),DVLG(1,1,IOA),896)
3055         SMALL OUT(DV(129),DVLG(1,2,IOA),896)
3064         SMALL OUT(DW(129),DVLG(1,3,IOA),896)
     C       CALCULATE ROWS 15 TO 30
3073         N2=769
3073         N3=1280
3074         N4=1
3075         NEXT=3
3076     *   M=3
3100         GO TO 613
```

```
3100    624 CALL DVDTMP(U,V,W,DU,DV,DW,M,CN,C8,C11,C12,ITIME,RMSD,RMSU,RMSV,
           2RMSW,SK1,SK2,UMAX,VMAX,WMAX,SPR,ISET)
3130        SMALL OUT(DU(1),DVLG(897,1,IOA),1024)
3140        SMALL OUT(DV(1),DVLG(897,2,IOA),1024)
3147        SMALL OUT(DW(1),DVLG(897,3,IOA),1024)
      C     CALCULATE ROWS 31 TO 46
3156        N2=1793
3156        N3=1280
3157        N4=1
3160        NEXT=4
3161        M=4
3163        GO TO 613
3163    625 CALL DVDTMP(U,V,W,DU,DV,DW,M,CN,C8,C11,C12,ITIME,RMSD,RMSU,RMSV,
           2RMSW,SK1,SK2,UMAX,VMAX,WMAX,SPR,ISET)
3213        SMALL OUT(DU(1),DVLG(1921,1,IOA),1024)
3223        SMALL OUT(DV(1),DVLG(1921,2,IOA),1024)
3232        SMALL OUT(DW(1),DVLG(1921,3,IOA),1024)
3241        I= MOD(IPLANE,2)
3244        IF(I.NE.0)GO TO 619
3246    606 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)+IDONE(W8)
3264        IF(I.NE.5)GO TO 606
3266        IF(IPLANE.EQ.64)GO TO 616
3270        IOB=ZM2-1
3271        I=IPLANE+5
3272        IF(I.GT.64)I=I-64
3275        NSECT=29*(I-1)
3300        IF(NSECT.GT.1512)NSECT=NSECT-1536
3304        IF(ITIME.LT.0)GO TO 631
3306        IF(ISET.EQ.0)GO TO 630
3307        GO TO(631,632,633,634)IM
3317    630 GO TO(632,633,634,631)IM
3327    631 I=IRANR (RQ1,VLG(1,1,IOB),24576,NSECT,W1)
3344        GO TO 616
3344    632 I=IRANR (RQ3,VLG(1,1,IOB),24576,NSECT,W3)
3361        GO TO 616
3361    633 I=IRANR (RQ2,VLG(1,1,IOB),24576,NSECT,W2)
3376        GO TO 616
3376    634 I=IRANR (RQ4,VLG(1,1,IOB),24576,NSECT,W4)
3413    616 IOB=IOA-1
3415        NSECT=IPLANE*24
3417        IF(NSECT.GT.1512)NSECT=NSECT-1536
3423        I=IRANW(RQ8,DVLG(1,1,IOB),24576,NSECT,W8)
3440    619 IOA=IOA+1
3442        IF(IOA.GE.5)IOA=1
3445        ZM2=ZM2+1
3446        IF(ZM2.GE.9)ZM2=1
3451    620 CONTINUE
3453    626 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)+IDONE(W8)
3471        IF(I.NE.5)GO TO 626
3473        IF(ITIME.LE.3)I=IRANR(RQ5,DLG(1),16384,0,D1)
3511        I=IRANR(RQ6,VLG(1),98304,0,W8)
3525    109 I=IDONE(W8)
3527        IF(I.NE.1)GO TO 109
3531        ZM2=1
3532        IOA=1
```

```
3533          DO 20 IPLANE=1,64
      C       TRANSFER ROWS 45 TO 64 TO CALCULATE 47 TO 62
3534          IF(ITIME.GT.3)GO TO 26
3537          SMALL IN(P(1),DLG(1,1,IOA),4096)
3546          GO TO 28
3546       26 DO 27 I=1,4096
3553       27 P(I)=0.
3555       28 CONTINUE
3555          N2=2817
3556          N3=1280
3557          N4=1
3560          NEXT=1
3561          M=1
3562       13 K=ZM2-1
3564          DO 21 Z=1,5
3566          K=K+1
3567          IF(K.GT.8)K=1
3572          SMALL IN(USM(1,N4,Z),VLG(N2,1,K),N3)
3606          SMALL IN(VSM(1,N4,Z),VLG(N2,2,K),N3)
3622       21 SMALL IN(WSM(1,N4,Z),VLG(N2,3,K),N3)
3640          GO TO(22,23,24,25)NEXT
3650       22 IF(ISET.EQ.0)C7=1./TDT
3653          CALL CALCPR(USM,VSM,WSM,P,M,C1,C7)
3663          C7=12./(5.*DELT)
3666          DO 14 J=1,4
3670          JJ=J+16
3671          DO 14 I=1,64
3673          DO 14 Z=1,5
3707          USM(I,J,Z)=USM(I,JJ,Z)
3711          VSM(I,J,Z)=VSM(I,JJ,Z)
3712       14 WSM(I,J,Z)=WSM(I,JJ,Z)
      C       CALCULATE ROWS 63,64 AND 1 TO 14
3720          N2=1
3720          N3=1024
3721          N4=5
3722          NEXT=2
3723          M=2
3724          GO TO 13
3725       23 IF(ISET.EQ.0)C7=1./TDT
3730          CALL CALCPR(USM,VSM,WSM,P,M,C1,C7)
3740          C7=12./(5.*DELT)
      C       CALCULATE ROWS 15 TO 30
3743          N2=769
3744          N3=1280
3745          N4=1
3746          NEXT=3
3747          M=3
3750          GO TO 13
3750       24 IF(ISET.EQ.0)C7=1./TDT
3753          CALL CALCPR(USM,VSM,WSM,P,M,C1,C7)
3763          C7=12./(5.*DELT)
      C       CALCULATE ROWS 31 TO 46
3766          N2=1793
3767          N3=1280
3770          N4=1
```

```
3771         NEXT=4
3772         M=4
3773         GO TO 13
3773      25 IF(ISET.EQ.0)C7=1./TDT
3776         CALL CALCPR(USM,VSM,WSM,P,M,C1,C7)
4006         C7=12./(5.*DELT)
4011         CALL FFTFXY(P)
4013         I=MOD(IOA,2)
4017         IF(I.NE.0)GO TO 16
4020         IOB=IOA-1
4022         SMALL OUT(P(1,1 ,1),PLG(1   ,2,IOB),2048)
4031         SMALL OUT(P(1,1 ,2),PLG(2049,2,IOB),2048)
4040         SMALL OUT(P(1,33,1),PLG(1   ,2,IOA),2048)
4047         SMALL OUT(P(1,33,2),PLG(2049,2,IOA),2048)
4056         GO TO 17
4056      16 IOB=IOA+1
4060         SMALL OUT(P(1,1 ,1),PLG(1   ,1,IOA),2048)
4067         SMALL OUT(P(1,1 ,2),PLG(2049,1,IOA),2048)
4076         SMALL OUT(P(1,33,1),PLG(1   ,1,IOB),2048)
4105         SMALL OUT(P(1,33,2),PLG(2049,1,IOB),2048)
4114      17 CONTINUE
          C  PAUSE IF THERE IS ANY OUTSTANDING I/O
4114         I=MOD(IPLANE,2)
4120         IF(I.NE.0)GO TO 19
4121     113 I=IDONE(D1)+IDONE(W8)
4126         IF(I.NE.2)GO TO 113
4130         IF(IPLANE.EQ.64)GO TO 116
4132         I=IPLANE+3
4133         IF(I.GT.64)I=I-64
4136         NSECT=8*(I-1)
          C  READ DIV OF PLANE+2 INTO LCM DIV(1,1,IOA)
4140         IOB=IOA-1
4142         IF(ITIME.LE.3)I=IRANR(RQ5,DLG(1,1,IOB),8192,NSECT,D1)
4162         I=IPLANE+5
4164         IF(I.GT.64)I=I-64
4167         NSECT=24*(I+1)
4172         IF(NSECT.GT.1512)NSECT=NSECT-1536
          C  READ VELOCITIES OF PLANE+4 INTO LCM VEL(4096,1,ZM2)
4176         IOB=ZM2-1
4200         IF(IOB.LE.0)STOP
4203         I=IRANR(RQ8,VLG(1,1,IOB),24576,NSECT,W8)
4220     116 I=IDONE(PUPR)
4222         IF(I.NE.1)GO TO 116
4224     117 I=IDONE(PLWR)
4226         IF(I.NE.1)GO TO 117
4230         NSECT=8*(IPLANE-2)
4232         IF(NSECT.GT.504)NSECT=NSECT-512
          C  WRITE(PLG(1,1,IOA) FROM LCM TO DISK
4236         IOB=IOA-1
4240         I=IRANW (RQ6,PLG(1,1 ,IOB),8192,NSECT,PUPR)
4253         I=IRANW (RQ7,PLG(1,1,IOA),8192,NSECT,PLWR)
4270     118 CONTINUE
4270      19 IOA=IOA+1
4272         IF(IOA.GE.5)IOA=1
4275         ZM2=ZM2+1
```

```
4276        IF(ZM2.GE.9)ZM2=1
4301     20 CONTINUE
      C
      C     ************      PHASE II   ***************
      C
      C     WAIT FOR OUTSTANDING I/O
4303    149 I=IDONE(PUPR)+IDONE(PLWR)+IDONE(D1)+IDONE(W1)+IDONE(W2)+IDONE(W3)+
           2IDONE(W4)+IDONE(W8)
4331        IF(I.NE.8)GO TO 149
      C     READ INTO PZ1,PZ2,PZ3, PRESSURES FOR I=1,64,J=1,32,Z=1,64
4333        I=IRANR (RQ6,PZ1(1,1,1),98304 ,0   ,PUPR)
4346    261 I=IDONE(PUPR)
4350        IF(I.NE.1)GO TO 261
4352        I=IRANR (RQ6,PZ2(1,1,1),98304 ,192,PUPR)
4365    262 I=IDONE(PUPR)
4367        IF(I.NE.1)GO TO 262
4371        I=IRANR (RQ6,PZ3(1,1,1),65536 ,384,PUPR)
      C     WAIT FOR OUTSTANDING I/O
4404    263 I=IDONE(PUPR)
4406        IF(I.NE.1)GO TO 263
4410        MM=0
4411     36 M=1
4412        DO 30 J=1,8
4414        JK=(J-1)*256+1
4417        DO 32 K=1,24
4421        KK=K+24
4422        SMALL IN(PZ(1,K ,1),PZ1(JK,1,K),256)
4433        SMALL IN(PZ(1,K ,2),PZ1(JK,2,K),256)
4443        SMALL IN(PZ(1,KK,1),PZ2(JK,1,K),256)
4454     32 SMALL IN(PZ(1,KK,2),PZ2(JK,2,K),256)
4466        DO 33 K=1,16
4467        KK=K+48
4470        SMALL IN(PZ(1,KK,1),PZ3(JK,1,K),256)
4501     33 SMALL IN(PZ(1,KK,2),PZ3(JK,2,K),256)
4512        CALL DELSQ(PZ,C,C64,S,S64,M,MM,C6)
4522        DO 34 K=1,24
4524        KK=K+24
4525        SMALL OUT(PZ(1,K ,1),PZ1(JK,1,K),256)
4536        SMALL OUT(PZ(1,K ,2),PZ1(JK,2,K),256)
4546        SMALL OUT(PZ(1,KK,1),PZ2(JK,1,K),256)
4557     34 SMALL OUT(PZ(1,KK,2),PZ2(JK,2,K),256)
4571        DO 35 K=1,16
4572        KK=K+48
4573        SMALL OUT(PZ(1,KK,1),PZ3(JK,1,K),256)
4604     35 SMALL OUT(PZ(1,KK,2),PZ3(JK,2,K),256)
4615     30 M=M+4
4621        IF(MM.EQ.32)GO TO 38
      C     WRITE PZ1,PZ2,PZ3 TO PRESSURES FOR I=1,64,J=1,32,Z=1,64
4623        I=IRANW (RQ6,PZ1(1,1,1),98304 ,0   ,PUPR)
4636    161 I=IDONE(PUPR)
4640        IF(I.NE.1)GO TO 161
4642        I=IRANW (RQ6,PZ2(1,1,1),98304 ,192,PUPR)
4655    162 I=IDONE(PUPR)
4657        IF(I.NE.1)GO TO 162
4661        I=IRANW (RQ6,PZ3(1,1,1),65536 ,384,PUPR)
```

```
      C     WAIT FOR OUTSTANDING I/O
4674    163 I=IDONE(PUPR)
4676        IF(I.NE.1)GO TO 163
      C     READ INTO PZ1,PZ2,PZ3, PRESSURES FOR I=1,64,J=33,64,Z=1,64
4700        I=IRANR (RQ7,PZ1(1,1,1),98304 ,0   ,PLWR)
4713    171 I=IDONE(PLWR)
4715        IF(I.NE.1)GO TO 171
4717        I=IRANR (RQ7,PZ2(1,1,1),98304 ,192,PLWR)
4732    172 I=IDONE(PLWR)
4734        IF(I.NE.1)GO TO 172
4736        I=IRANR (RQ7,PZ3(1,1,1),65536 ,384,PLWR)
      C     WAIT FOR OUTSTANDING I/O
4751    173 I=IDONE(PLWR)
4753        IF(I.NE.1)GO TO 173
4755        MM=32
4756        GO TO 36
4757     38 CONTINUE
      C     WRITE PZ1,PZ2,PZ3 TO PRESSURES FOR I=1,64,J=1,32,Z=1,64
4757        I=IRANW (RQ7,PZ1(1,1,1),98304 ,0   ,PLWR)
4773    181 I=IDONE(PLWR)
4775        IF(I.NE.1)GO TO 181
4777        I=IRANW (RQ7,PZ2(1,1,1),98304 ,192,PLWR)
5012    182 I=IDONE(PLWR)
5014        IF(I.NE.1)GO TO 182
5016        I=IRANW (RQ7,PZ3(1,1,1),65536 ,384,PLWR)
      C     WAIT FOR OUTSTANDING I/O
5031    183 I=IDONE(PLWR)
5033        IF(I.NE.1)GO TO 183
5035    191 I=IDONE(D1)*IDONE(W1)*IDONE(W2)*IDONE(W3)*IDONE(W4)
5053        IF(I.NE.5)GO TO 191
      C
      C     ************      PHASE III     ***************
      C
5055        ZM2=1
5055        IOA=1
5056    150 I=IDONE(PUPR)*IDONE(PLWR)*IDONE(D1)*IDONE(W1)*IDONE(W2)*IDONE(W3)*
           2IDONE(W4)*IDONE(W8)
5104        IF(I.NE.8)GO TO 150
      C     READ PHAT OF PLANES 63 TO 4 INTO PHAT(128,64,6) IN LCM
      C     READ VELOCITIES OF PLANES 63 TO 4 INTO VLG(4096,3,6) IN LCM
      C     READ OLD VELOCITIES OF PLANES 1,2 INTO VNM1(4096,3,2) IN LCM
5106        I=IRANR(RQ8,VLG(1),49152,48,W8)
5121    256 I=IDONE(W8)
5123        IF(I.NE.1)GO TO 256
5125        NSECT=4096
5126        DO 205 Z=1,6
5130    203 I=IDONE(PUPR)
5132        IF(I.NE.1)GO TO 203
5134    204 I=IDONE(PLWR)
5136        IF(I.NE.1)GO TO 204
5140        I=IRANR (RQ6,PHAT(1,1 ,Z),4096,NSECT,PUPR)
5154        I=IRANR (RQ7,PHAT(1,2 ,Z),4096,NSECT,PLWR)
5170        NSECT=NSECT+8
5172        IF(NSECT.GE.512)NSECT=NSECT-512
5175    205 CONTINUE
```

```
5177          DO 40 Z=1,5
5201          SMALL IN(PA(1    ,1),PHAT(1    ,1,Z),2048)
5210          SMALL IN(PA(2049,1),PHAT(1    ,2,Z),2048)
5217          SMALL IN(PA(1    ,2),PHAT(2049,1,Z),2048)
5226          SMALL IN(PA(2049,2),PHAT(2049,2,Z),2048)
5235          CALL FFTBXY(PA)
      C       FFTBXY LEAVES THE REAL PRESSURE IN PR(64,64) WHICH IS EQUIVALENCED
      C       TO WOLD(1,1,1)
5236          SMALL OUT(PA(1)   ,PC(1,1,Z),4096)
5246       40 CONTINUE
      C       LAST DIMENSION FOR PC IS 6
5250          IOB=1
5251          DO 50 IPLANE=1,64
5252          M=1
      C       TRANSFER ROWS 45 TO 64 TO CALCULATE 47 TO 62
5253          N1=45
5253          N2=2817
5254          N3=1280
5255          N4=1
5256          NEXT=1
5257       60 K=ZM6-1
5261          DO 61 Z=1,5
5263          K=K+1
5264          IF(K.GT.6)K=1
5267       61 SMALL IN(PD(1,N4,Z),PC(1,N1,K),N3)
5306          SMALL IN(U(1,N4,3),VLG(N5,1,IOB),N3)
5320          SMALL IN(V(1,N4,3),VLG(N2,2,IOB),N3)
5330          SMALL IN(W(1,N4,3),VLG(N2,3,IOB),N3)
5340          GO TO (62,63,64,65)NEXT
5350       62 CALL DVDT(U,V,W,DU,DV,DW,PD,M,CN,C8,C11,RMSU,RMSV,RMSW)
5367          N1=2945
5370          N2=1024
5371          N3=1
5372       69 SMALL OUT(DU(N3),DULG(N1,1,IOA),N2)
5404          SMALL OUT(DV(N3),DULG(N1,2,IOA),N2)
5413          SMALL OUT(DW(N3),DULG(N1,3,IOA),N2)
5422          GO TO(72,73,74,75,76)NEXT
5433       72 DO 52 J=1,4
5435          JJ=J+16
5436          DO 52 I=1,64
5440          DO 52 Z=1,5
5454          U(I,J,Z)=U(I,JJ,Z)
5456          V(I,J,Z)=V(I,JJ,Z)
5457          W(I,J,Z)=W(I,JJ,Z)
5460       52 PD(I,J,Z)=PD(I,JJ,Z)
      C       CALCULATE ROWS 63,64 AND 1 TO 14
5466          N4=5
5466          N1=1
5467          N2=1
5470          N3=1024
5471          NEXT=2
5473          GO TO 60
5473       63 CALL DVDT(U,V,W,DU,DV,DW,PD,M,CN,C8,C11,RMSU,RMSV,RMSW)
5512          N1=3969
5513          N2=128
```

```
5514          N3=1
5515          GO TO 69
5516       73 NEXT=3
5517          N1=1
5520          N2=896
5521          N3=129
5522          GO TO 69
5523       74 N1=13
           C     CALCULATE ROWS 15 TO 30
5524          N2=769
5525          N3=1280
5526          N4=1
5527          GO TO 60
5530       64 CALL DVDT(U,V,W,DU,DV,DW,PD,M,CN,CB,C11,RMSU,RMSV,RMSW)
5547          NEXT=4
5550          N1=897
5551          N2=1024
5552          N3=1
5553          GO TO 69
5554       75 N1=29
           C     CALCULATE ROWS 31 TO 46
5555          N2=1793
5556          N3=1280
5557          N4=1
5560          GO TO 60
5561       65 CALL DVDT(U,V,W,DU,DV,DW,PD,M,CN,CB,C11,RMSU,RMSV,RMSW)
5600          NEXT=5
5601          N1=1921
5602          N2=1024
5603          N3=1
5604          GO TO 69
5605       76 CONTINUE
           C     WAIT FOR ANY OUTSTANDING I/O
           C     WRITE DU,DV,DW TO DISK
           C     READ PHAT OF IPLANE+4 TO PHATA(IOA)
           C     READVLG OF IPLANE+4 TO VLG(ZM2)
5605     210 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)+IDONE(W8)
5623          IF(I.NE.5)GO TO 210
5625          NSECT=24*(IPLANE+1)
5627          IF(NSECT.GT.1512)NSECT=NSECT-1536
5633          IF(ITIME.LT.0)GO TO 223
5635          IF(ISET.EQ.0)GO TO 211
5636          GO TO(221,222,223,224)IM
5646       211 GO TO (224,221,222,223)IM
5656       221 I=IRANW(RQ1,DULG(1,1,IOA),12288,NSECT,W1)
5673          GO TO 220
5673       222 I=IRANW(RQ3,DULG(1,1,IOA),12288,NSECT,W3)
5710          GO TO 220
5710       223 I=IRANW(RQ2,DULG(1,1,IOA),12288,NSECT,W2)
5725          GO TO 220
5725       224 I=IRANW(RQ4,DULG(1,1,IOA),12288,NSECT,W4)
5742       220 IF(MOD(IOB,2).NE.0)GO TO 215
5746          IF(IPLANE.EQ.64)GO TO 50
5750          IOC=IOB-1
5751          NSECT=(IPLANE+4)*24
```

```
5754        IF(NSECT.GT.1512)NSECT=NSECT-1536
5760        I=IRANR(RQ8,VLG(1,1,IOC),24576,NSECT,W8)
5775    215 NSECT=8*(IPLANE+3)
5777        IF(NSECT.GT.504)NSECT=NSECT-512
6003    217 I=IDONE(PUPR)
6005        IF(I.NE.1)GO TO 217
6007    218 I=IDONE(PLWR)
6011        IF(I.NE.1)GO TO 218
6013        I=IRANR (RQ6,PHATA(1,1 ,IOA),4096,NSECT,PUPR)
6027        I=IRANR (RQ7,PHATA(1,2 ,IOA),4096,NSECT,PLWR)
6044    219 IOA=IOA+1
6046        IF(IPLANE.EQ.64)GO TO 50
6050        IF(IOA.GT.2)IOA=1
6053        IOB=IOB+1
6055        IF(IOB.GE.5)IOB=1
6060        SMALL IN(PA(1,1),PHATA(1,1,IOA),2048)
6070        SMALL IN(PA(2049,1),PHATA(1,2,IOA),2048)
6077        SMALL IN(PA(1,2),PHATA(2049,1,IOA),2048)
6106        SMALL IN(PA(2049,2),PHATA(2049,2,IOA),2048)
      C     EQUIVALENCE PHATA(1,1,5),PHATA(1,1,1)
6115        CALL FFTBXY(PA)
6116        ZP3=ZM2+5
6120        IF(ZP3.GT.6)ZP3=ZP3-6
6123        SMALL OUT(PA(1)   ,PC(1,1,ZP3),4096)
6133        ZM2=ZM2+1
6134        IF(ZM2.GE.7)ZM2=1
6137     50 CONTINUE
      C
      C
      C     ***************  PHASE IV   **************
      C
      C
6141    407 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)+IDONE(W8)+IDONE(D1)+IDON
           2E(PUPR)+IDONE(PLWR)
6167        IF(I.NE.8)GO TO 407
6171        IF(ITIME.LT.0)GO TO 1000
6173        IF(ISET.NE.0)GO TO 408
6174        ISET=1
6175        GO TO 1001
6175    408 GO TO (401,402,403,404)IM
6205    401 I=IRANR(RQ2,UTNM1(1),24576,0,W2)
6220        I=IRANR(RQ4,UTN  (1),24576, ,W4)
6233        I=IRANR(RQ3,UN   (1),24576,0,W3)
6246        I=IRANR(RQ1,UTNP1(1),24576, ,W1)
6262        GO TO 405
6262    402 I=IRANR(RQ4,UTNM1(1),24576,0,W4)
6275        I=IRANR(RQ1,UTN  (1),24576,0,W1)
6310        I=IRANR(RQ2,UN   (1),24576,0,W2)
6323        I=IRANR(RQ3,UTNP1(1),24576, ,W3)
6337        GO TO 405
6337    403 I=IRANR(RQ1,UTNM1(1),24576,0,W1)
6352        I=IRANR(RQ3,UTN  (1),24576,0,W3)
6365        I=IRANR(RQ4,UN   (1),24576,0,W4)
6400        I=IRANR(RQ2,UTNP1(1),24576,0,W2)
6414        GO TO 405
```

```
6414    404 I=IRANR(RQ3,UTNM1(1),24576,0,W3)
6427        I=IRANR(RQ2,UTN  (1),24576,0,W2)
6442        I=IRANR(RQ1,UN   (1),24576,0,W1)
6455        I=IRANR(RQ4,UTNP1(1),24576,0,W4)
6471    405 CONTINUE
6471    406 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)+IDONE(W8)
6507        IF(I.NE.5)GO TO 406
6511        GO TO(411,412,413,414)IM
6521    411 I=IRANR(RQ2,WTNM1(1),24576,48,W2)
6534        I=IRANR(RQ4,WTN  (1),24576,48,W4)
6547        I=IRANR(RQ3,WN   (1),24576,48,W3)
6562        I=IRANR(RQ1,WTNP1(1),24576,48,W1)
6576        GO TO 415
6576    412 I=IRANR(RQ4,WTNM1(1),24576,48,W4)
6611        I=IRANR(RQ1,WTN  (1),24576,48,W1)
6624        I=IRANR(RQ2,WN   (1),24576,48,W2)
6637        I=IRANR(RQ3,WTNP1(1),24576,48,W3)
6653        GO TO 415
6653    413 I=IRANR(RQ1,WTNM1(1),24576,48,W1)
6666        I=IRANR(RQ3,WTN  (1),24576,48,W3)
6701        I=IRANR(RQ4,WN   (1),24576,48,W4)
6714        I=IRANR(RQ2,WTNP1(1),24576,48,W2)
6730        GO TO 415
6730    414 I=IRANR(RQ3,WTNM1(1),24576,48,W3)
6743        I=IRANR(RQ2,WTN  (1),24576,48,W2)
6756        I=IRANR(RQ1,WN   (1),24576,48,W1)
6771        I=IRANR(RQ4,WTNP1(1),24576,48,W4)
7005    415 CONTINUE
7005        DO 450 N=1,32
7007        IF(MOD(N,2).EQ.J)GO TO 430
7013        IF(N.EQ.1)GO TO 417
7014    416 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)+IDONE(W8)
7032        IF(I.NE.5)GO TO 416
7034        J=0
7034        NSECT=(N-2)*48
7040        GO TO(421,422,423,424)IM
7050    421 I=IRANW(RQ2,WN   (1),24576,NSECT,W2)
7064        GO TO 425
7064    422 I=IRANW(RQ4,WN   (1),24576,NSECT,W4)
7100        GO TO 425
7100    423 I=IRANW(RQ1,WN   (1),24576,NSECT,W1)
7114        GO TO 425
7114    424 I=IRANW(RQ3,WN   (1),24576,NSECT,W3)
7130        GO TO 425
7130    417 J=1
7131    425 DO 429 M=1,6
7133        K=(M-1)*4096+1
7136        SMALL IN(VTNM1(1),UTNM1(K),4096)
7145        SMALL IN(VTN  (1),UTN  (K),4096)
7153        SMALL IN(VTNP1(1),UTNP1(K),4096)
7161        SMALL IN(VN   (1),UN   (K),4096)
7167        CALL ADVNC(TDT,M,CRMSU,CRMSV,CRMSW,CUMAX,CVMAX,CWMAX)
7177        SMALL OUT(VN(1),JN(K),4096)
7206    428 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)+IDONE(W8)
7224        IF(((I.NE.5).AND.(J.EQ.0)).AND.(M.EQ.6))GO TO 428
```

```
7236        IF((I.NE.5).OR.(J.EQ.1))GO TO 429
7245        J=1
7245        NSECT=N*48
7250        GO TO (431,432,433,434)IM
7260    431 I=IRANR(RQ2,WTNM1(1),24576,NSECT,W2)
7273        I=IRANR(RQ4,WTN  (1),24576,NSECT,W4)
7306        I=IRANR(RQ3,WN   (1),24576,NSECT,W3)
7321        I=IRANR(RQ1,WTNP1(1),24576,NSECT,W1)
7335        GO TO 427
7335    432 I=IRANR(RQ4,WTNM1(1),24576,NSECT,W4)
7356        I=IRANR(RQ1,WTN  (1),24576,NSECT,W1)
7363        I=IRANR(RQ2,WN   (1),24576,NSECT,W2)
7376        I=IRANR(RQ3,WTNP1(1),24576,NSECT,W3)
7412        GO TO 427
7412    433 I=IRANR(RQ1,WTNM1(1),24576,NSECT,W1)
7425        I=IRANR(RQ3,WTN  (1),24576,NSECT,W3)
7440        I=IRANR(RQ4,WN   (1),24576,NSECT,W4)
7453        I=IRANR(RQ2,WTNP1(1),24576,NSECT,W2)
7467        GO TO 427
7467    434 I=IRANR(RQ3,WTNM1(1),24576,NSECT,W3)
7502        I=IRANR(RQ2,WTN  (1),24576,NSECT,W2)
7515        I=IRANR(RQ1,WN   (1),24576,NSECT,W1)
7530        I=IRANR(RQ4,WTNP1(1),24576,NSECT,W4)
7544    427 CONTINUE
7544    429 CONTINUE
7546        GO TO 450
7547    430 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)+IDONE(W8)
7565        IF(I.NE.5)GO TO 430
7567        J=0
7567        NSECT=(N-2)*48
7573        GO TO(441,442,443,444)IM
7603    441 I=IRANW(RQ2,UN   (1),24576,NSECT,W2)
7617        GO TO 446
7617    442 I=IRANW(RQ4,UN   (1),24576,NSECT,W4)
7633        GO TO 446
7633    443 I=IRANW(RQ1,UN   (1),24576,NSECT,W1)
7647        GO TO 446
7647    444 I=IRANW(RQ3,UN   (1),24576,NSECT,W3)
7663    446 CONTINUE
7663    445 DO 459 M=1,6
7665        K=(M-1)*4096+1
7670        SMALL IN(VTNM1(1),WTNM1(K),4096)
7677        SMALL IN(VTN  (1),WTN  (K),4096)
7705        SMALL IN(VTNP1(1),WTNP1(K),4096)
7713        SMALL IN(VN   (1),WN   (K),4096)
7721        CALL ADVNC(TDT,M,CRMSU,CRMSV,CRMSW,CUMAX,CVMAX,CWMAX)
7731        SMALL OUT(VN(1),WN(K),4096)
7740    448 I=IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)+IDONE(W8)
7756        IF(((I.NE.5).AND.(J.EQ.0)).AND.(M.EQ.6))GO TO 448
7770        IF((I.NE.5).OR.(J.EQ.1))GO TO 459
7777        J=1
7777        IF(N.EQ.32)GO TO 459
10002       NSECT=N*48
10004       GO TO (451,452,453,454)IM
10014    451 I=IRANR(RQ2,UTNM1(1),24576,NSECT,W2)
```

```
10027         I=IRANR(RQ4,UTN   (1),24576,NSECT,W4)
10042         I=IRANR(RQ3,UN    (1),24576,NSECT,W3)
10055         I=IRANR(RQ1,UTNP1(1),24576,NSECT,W1)
10071         GO TO 457
10071    452  I=IRANR(RQ4,UTNM1(1),24576,NSECT,W4)
10104         I=IRANR(RQ1,UTN   (1),24576,NSECT,W1)
10117         I=IRANR(RQ2,UN    (1),24576,NSECT,W2)
10132         I=IRANR(RQ3,UTNP1(1),24576,NSECT,W3)
10146         GO TO 457
10146    453  I=IRANR(RQ1,UTNM1(1),24576,NSECT,W1)
10161         I=IRANR(RQ3,UTN   (1),24576,NSECT,W3)
10174         I=IRANR(RQ4,UN    (1),24576,NSECT,W4)
10207         I=IRANR(RQ2,UTNP1(1),24576,NSECT,W2)
10223         GO TO 457
10223    454  I=IRANR(RQ3,UTNM1(1),24576,NSECT,W3)
10236         I=IRANR(RQ2,UTN   (1),24576,NSECT,W2)
10251         I=IRANR(RQ1,UN    (1),24576,NSECT,W1)
10264         I=IRANR(RQ4,UTNP1(1),24576,NSECT,W4)
10300    457  CONTINUE
10300    459  CONTINUE
10302    450  CONTINUE
10304         GO TO(511,512,513,514)IM
10314    511  I=IRANW(RQ2,WN    (1),24576,1488,W2)
10330         GO TO 515
10330    512  I=IRANW(RQ4,WN    (1),24576,1488,W4)
10344         GO TO 515
10344    513  I=IRANW(RQ1,WN    (1),24576,1488,W1)
10360         GO TO 515
10360    514  I=IRANW(RQ3,WN    (1),24576,1488,W3)
10374    515  CONTINUE
10374   1000  ITIME=ITIME+1
10376         IF(ITIME.EQ.0)GO TO 1001
10377         ISET=0
10377         RMSD=SQRT(RMSD)/512.
10403         RMSU=SQRT(RMSU)/512.
10407         RMSV=SQRT(RMSV)/512.
10413         RMSW=SQRT(RMSW)/512.
10417         UMAX=SQRT(UMAX)
10421         VMAX=SQRT(VMAX)
10423         WMAX=SQRT(WMAX)
10425         CRMSU=SQRT(CRMSU)/512.
10431         CRMSV=SQRT(CRMSV)/512.
10435         CRMSW=SQRT(CRMSW)/512.
10441         CUMAX=SQRT(CUMAX)
10443         CVMAX=SQRT(CVMAX)
10445         CWMAX=SQRT(CWMAX)
10447         SK1=SK1/SQRT(SK2**3)*128.
10456         PRINT 899,RMSD,RMSU,RMSV,RMSW,SK1
          899 FORMAT(*  RMSD=*E14.7*  RMSU=*E14.7*  RMSV=*E14.7*  RMSW=*E14.7*
             2SKEWNESS=*E14.7)
10473         PRINT 888,UMAX,VMAX,WMAX
          888 FORMAT(*  UMAX=*E14.7*  VMAX=*E14.7*  WMAX=*E14.7)
10505         PRINT 895,CRMSU,CRMSV,CRMSW
10517         PRINT 888,CUMAX,CVMAX,CWMAX
          895 FORMAT(*          RMSU=*E14.7*  RMSV=*E14.7*  RMSW=*E14.7)
```

```
10531         XX=20./(3.1415927*64.*15360.)
10533         DO 889 I=1,64
10541         SPI(I)=0.
10542     889 SPR(I)=XX*SPR(I)
10543         CALL FFT2(SPR(1),SPI(1),64,-1)
10546         PRINT 887
          887 FORMAT(*  E11=*)
10552         PRINT 886,(SPR(I),I=1,32)
          886 FORMAT(1X,8E12.5)
10560         CALL SECOND(TX)
10562         PRINT 1900,TX
         1900 FORMAT(*  CPU TIME =*E14.7)
10570         GO TO(1500,1200,1600,1300)IM
10600    1200 CONTINUE
10600         IF(IDONE(W4).NE.1)GO TO 1200
          C   SAVE TIME STEP 3 FROM W4 IN FSET7
10604         DO 1210 K=1,8
10605         NSECT=(K-1)*192
10607         I=IRANR(RQ3,DUM1(1),98304,NSECT,W4)
10623    1201 I=IDONE(W4)
10625         IF(I.NE.1)GO TO 1201
10627         DO 1210 J=1,3
10631         JK=(J-1)*32768+1
10634         SMALL IN(DUMSM(1),DUM1(JK),32768)
10643         WRITE(7)(DUMSM(I),I=1,32768)
10650    1210 CONTINUE
10654         M=(ITIME-40)/2
10657         END FILE 7
10661         GO TO(1401,1402,1403,1404,1405,1406,1407,1408)M
10675    1300 CONTINUE
10675         IF(IDONE(W3).NE.1)GO TO 1300
          C   SAVE TIME STEP  5 FROM W3 IN FSET7
10701         DO 1310 K=1,8
10702         NSECT=(K-1)*192
10704         I=IRANR(RQ3,DUM1(1),98304,NSECT,W3)
10720    1301 I=IDONE(W3)
10722         IF(I.NE.1)GO TO 1301
10724         DO 1310 J=1,3
10726         JK=(J-1)*32768+1
10731         SMALL IN(DUMSM(1),DUM1(JK),32768)
10740         WRITE(7)(DUMSM(I),I=1,32768)
10745    1310 CONTINUE
10751         M=(ITIME-40)/2
10754         END FILE 7
10756         GO TO(1401,1402,1403,1404,1405,1406,1407,1408)M
10772    1401 CALL AFSREL(5LFSET7,0,I1TAPE)
10775         GO TO 1269
10776    1402 CALL AFSREL(5LFSET7,0,I2TAPE)
11001         GO TO 1269
11002    1403 CALL AFSREL(5LFSET7,0,I3TAPE)
11005         GO TO 1269
11006    1404 CALL AFSREL(5LFSET7,0,I4TAPE)
11011         GO TO 1269
11012    1405 CALL AFSREL(5LFSET7,0,I5TAPE)
11015         GO TO 1269
```

```
11016   1406 CALL AFSREL(5LFSET7,0,I6TAPE)
11021        GO TO 1269
11022   1407 CALL AFSREL(5LFSET7,0,I7TAPE)
11025        GO TO 1269
11026   1408 CALL AFSREL(5LFSET7,0,I8TAPE)
11031        GO TO 1269
11032   1500 CONTINUE
11032        IF(IDONE(W2).NE.1)GO TO 1500
11036        CALL OPEN(5LFSET7,0,234000B,0,0,0,3200,ITAPE)
11046        DO 1510 K=1,8
11050        NSECT=(K-1)*192
11052        I=IRANR(RQ2,DUM1(1),98304,NSECT,W2)
11066   1501 I=IDONE(W2)
11070        IF(I.NE.1)GO TO 1501
11072        DO 1510 J=1,3
11074        JK=(J-1)*32768+1
11077        SMALL IN(DUMSM(1),DUM1(JK),32768)
11106        WRITE(7)(DUMSM(I),I=1,32768)
11113   1510 CONTINUE
11117        GO TO 1269
11120   1600 CONTINUE
11120        IF(IDONE(W1).NE.1)GO TO 1600
11124        CALL OPEN(5LFSET7,0,234000B,0,0,0,3200,ITAPE)
11134        DO 1610 K=1,8
11136        NSECT=(K-1)*192
11140        I=IRANR(RQ1,DUM1(1),98304,NSECT,W1)
11154   1601 I=IDONE(W1)
11156        IF(I.NE.1)GO TO 1601
11160        DO 1610 J=1,3
11162        JK=(J-1)*32768+1
11165        SMALL IN(DUMSM(1),DUM1(JK),32768)
11174        WRITE(7)(DUMSM(I),I=1,32768)
11201   1610 CONTINUE
11205   1269 IF(ITIME.LT.52)GO TO 1001
11210   1100 I=IDONE(D1)+IDONE(W1)+IDONE(W2)+IDONE(W3)+IDONE(W4)+IDONE(W8)
            2+IDONE(PUPR)+IDONE(PLWR)
11236        IF(I.NE.8)GO TO 1100
11240        CALL OPEN(5LFSET5,0,234000B,0,0,0,3200,ITAPE)
11250        DO 1120 K=1,8
11252        NSECT=(K-1)*192
11254        I=IRANR(RQ1,DUM1(1),98304,NSECT,W1)
11270   1102 I=IDONE(W1)
11272        IF(I.NE.1)GO TO 1102
11274        DO 1120 J=1,3
11276        JK=(J-1)*32768+1
11301        SMALL IN(DUMSM(1),DUM1(JK),32768)
11310        WRITE(5)(DUMSM(I),I=1,32768)
11315   1120 CONTINUE
11321        DO 1110 K=1,8
11323        NSECT=(K-1)*192
11325        I=IRANR(RQ2,DUM1(1),98304,NSECT,W2)
11341   1101 I=IDONE(W2)
11343        IF(I.NE.1)GO TO 1101
11345        DO 1110 J=1,3
11347        JK=(J-1)*32768+1
```

```
11352          SMALL IN(DUMSM(1),DUM1(JK),32768)
11361          WRITE(5)(DUMSM(I),I=1,32768)
11366    1110  CONTINUE
11372          END FILE 5
11374          CALL AFSREL(5LFSET5,0,I05TAPE)
11377          CALL OPEN(5LFSET6,0,23400B,0,0,0,3200,ITAPE)
11407          DO 1140 K=1,8
11411          NSECT=(K-1)*192
11413          I=IRANR(RQ3,DUM1(1),98304,NSECT,W3)
11427    1104  I=IDONE(W3)
11431          IF(I.NE.1)GO TO 1104
11433          DO 1140 J=I,3
11435          JK=(J-1)*32768+1
11440          SMALL IN(DUMSM(1),DUM1(JK),32768)
11447          WRITE(6)(DUMSM(I),I=1,32768)
11454    1140  CONTINUE
11460          DO 1130 K=1,8
11462          NSECT=(K-1)*192
11464          I=IRANR(RQ4,DUM1(1),98304,NSECT,W4)
11500    1103  I=IDONE(W4)
11502          IF(I.NE.1)GO TO 1103
11504          DO 1130 J=1,3
11506          JK=(J-1)*32768+1
11511          SMALL IN(DUMSM(1),DUM1(JK),32768)
11520          WRITE(6)(DUMSM(I),I=1,32768)
11525    1130  CONTINUE
11531          END FILE 6
11533          CALL AFSREL(5LFSET6,0,I06TAPE)
11536          DO 1131 I=1,5
11544          I05TAPE(I)=I07TAPE(I)
11545    1131  I06TAPE(I)=I08TAPE(I)
11547          IF(ITIME.EQ.24)GO TO 1001
11551          STOP
11553          END
```

PROGRAM LENGTH INCLUDING I/O REQUEST TABLES = GAP
15131

STATEMENT ASSIGNMENTS

STMT NO.	LOCATION	STMT NO.	LOCATION	STMT NO.	LOCATION	STMT NO
13	3563	16	4057	17	4115	19
20	4302	22	3651	23	3726	24
25	3774	26	3547	28	3556	36
38	4760	50	6140	60	5260	61
62	5351	63	5474	64	5531	65
69	5373	72	5434	73	5517	74
75	5555	76	5606	100	2406	109
113	4122	116	4221	117	4225	118
130	1173	131	1220	132	1247	133
134	1325	139	1354	149	4304	150
151	1441	152	1455	153	1471	154
155	1521	156	1523	157	1540	161
162	4656	163	4675	171	4714	172

```
      SUBROUTINE ADVNC(TDT,M,RMSU,RMSV,RMSW,UMAX,VMAX,WMAX)
20    COMMON U(32768)
C     C1=1./(2.*DELT)
20    C1=1./TDT
C     C2=1./(2.*DELT**2)
21    C2=2./TDT**2
C     C3=DELT
23    C3=.5*TDT
C     C4=DELT**2/2.
25    C4=.125*TDT**2
C     C5=DELT**3/6.
27    C5=TDT**3/24.
30    C6=C4*C1
32    C7=C5*C2
35    DO 10 I=1,4096
43    U(I)=U(I)+C3*U(I+8192 )+C6*(U(I+12288)-U(I+4096))+C7*(U(I+12288)-
     22.*U(I+8192 )+U(I+4096))
53  10 CONTINUE
54    GO TO(20,30,40,20,30,40)M
71  20 DO 25 I=1,4096
73    X=U(I)**2
75    RMSU=RMSU+X
76    IF(X.GT.UMAX)UMAX=X
101 25 CONTINUE
103   GO TO 50
104 30 DO 35 I=1,4096
106   X=U(I)**2
110   RMSV=RMSV+X
111   IF(X.GT.VMAX)VMAX=X
114 35 CONTINUE
116   GO TO 50
117 40 DO 45 I=1,4096
121   X=U(I)**2
123   RMSW=RMSW+X
124   IF(X.GT.WMAX)WMAX=X
127 45 CONTINUE
131 50 RETURN
132   END
```

SUBPROGRAM LENGTH - ADVNC
162

STATEMENT ASSIGNMENTS

STMT NO.	LOCATION	STMT NO.	LOCATION	STMT NO.	LOCATION	STMT NO
20	72	25	102	30	105	35
40	120	45	130	50	132	

BLOCK NAMES AND LENGTHS

	100000

VARIABLE ASSIGNMENTS

NAME	LOCATION	NAME	LOCATION	NAME	LOCATION	NAME

```
      SUBROUTINE DVRGNC(C12,RMSD,RMSU,RMSV,RMSW,SK1,SK2,UMAX,VMAX,WMAX,M
     2,ID,DELT)
25    LCM/BB1/VLG
25    LCM/BB2/WLG
25    LCM/BB3/DLG
25    DIMENSION VLG(4096,3,8),WLG(4096,3,8),DLG(64,64,16),A(20480),D(819
     22)
25    COMMON DUMSM(32768)
25    EQUIVALENCE (A(1),DUMSM(1)),(D(1),DUMSM(20481))
C     DISPLAY OFF
25    N=MOD(M,4)
31    IF(N.EQ.0)N=4
33    OTDT=DELT/12.
35    TTDT=2.*DELT/3.
40    DO 5, K=3,6
41    IF((N.EQ.2).OR.(N.EQ.4))GO TO 2
50    SMALL IN(A(3),VLG(1,1,K),4096)
57    GO TO 3
57  2 SMALL IN(A(3),WLG(1,1,K),4096)
67  3 A(1)=A(4097)
71    A(2)=A(4098)
72    A(4099)=A(3)
74    A(4100)=A(4)
75    DO 10 I=1,4096
104  10 D(I)=     A(I)-A(I+4)+8.*(A(I+3)-A(I+1))
111   IF((N.EQ.2).OR.(N.EQ.4))GO TO 12
124   SMALL IN(A(129),VLG(1,2,K),4096)
133   GO TO 14
133  13 SMALL IN(A(129),WLG(1,2,K),4096)
143  14 DO 15 I=1,128
150  15 A(I)=A(I+4096)
152     DO 16 I=4225,4352
162  16 A(I)=A(I-4096)
164     DO 20 I=1,4096
176  20 D(I)=D(I)+     A(I)-A(I+256)+8.*(A(I+192)-A(I+64))
204     J=K-6
205     JJ=1
206     IF((N.EQ.2).OR.(N.EQ.4))GO TO 23
222     DO 25 I=1,5
223     SMALL IN(A(JJ),VLG(1,3,J),4096)
233     J=J+1
234  25 JJ=JJ+4096
237     GO TO 24
237  23 DO 26 I=1,5
241     SMALL IN(A(JJ),WLG(1,3,J),4096)
251     J=J+1
252  26 JJ=JJ+4096
255  24 DO 30 I=1,4096
263  30 D(I)=C12*(D(I)+A(I)-A(I+16384)+8.*(A(I+12288)-A(I+4096)))
271     J=K-6+(N-1)*4
275     GO TO(40,32,33)ID
310  32 SMALL IN(D(4097),DLG(1,1,J),4096)
320     DO 36 I=1,4096
325     D(I)=-OTDT*D(I)+D(I+4096)
327  36 CONTINUE
```

```
330         GO TO 40
334      33 SMALL IN(D(4097),DLG(1,1,J),4096)
344         DO 37 I=1,4096
351         D(I)=TTDT*D(I)+D(I+4096)
353      37 CONTINUE
357      40 SMALL OUT(D(1),DLG(1,1,J),4096)
367      50 CONTINUE
371         RETURN
372         END
```

SUBPROGRAM LENGTH - DVRGNC

424

STATEMENT ASSIGNMENTS

STMT NO	LOCATION	STMT NO	LOCATION	STMT NO	LOCATION	STMT NO
2	60	3	70	13	134	14
23	240	24	256	32	311	33
40	360					

BLOCK NAMES AND LENGTHS

	100000

VARIABLE ASSIGNMENTS

NAME		LOCATION	NAME		LOCATION	NAME		LOCATION	NAME	
A	R	0C01	C12	R	0	D	R	50000C01	DELT	
DUMSM	R	0C01	I	I	415	ID	I	13	J	
JJ	I	417	K	I	420	M	I	12	N	
OTDT	R	422	RMSD	R	1	RMSU	R	2	RMSV	
RMSW	R	4	SK_1	R	5	SK_2	R	6	TTDT	
UMAX	R	7	VMAX	R	10	WMAX	R	11		

LCM BLOCK NAMES AND LENGTHS

BB_1	300000	BB2	300000	BB3	200000

LCM VARIABLE ASSIGNMENTS

NAME		LOCATION	NAME		LOCATION	NAME		LOCATION
DLG	R	0L03	VLG	R	0L01	WLG	R	0L02

EXTERNAL ASSIGNMENTS

ACGOER R

START OF	CONSTANTS	TEMPORARIES	INDIRECTS	UNUSED COMPI
	374	403	413	733

```
      FUN-LCM89                          74/11/14    16.32.37    V5CLARK1HA   PAGE NO. 1

                SUBROUTINE FFTFXY(P)
      3         DIMENSION P(4096,2)
      3         DO 1 I=1,4096
      7       1 P(I,2)=0.
     11         CALL FFT2(P(1,1),P(1,2),4096,-1)
     15         RETURN
     16         END


SUBPROGRAM LENGTH - FFTFXY
    30

VARIABLE ASSIGNMENTS
NAME    *  LOCATION            NAME    *  LOCATION

1       *I      27              P       *R       0

EXTERNAL ASSIGNMENTS
FFT2    *R

START OF      -       CONSTANTS          TEMPORARIES          INDIRECTS      -       UNUSED COMPI
                          20                   22                   24                   746
```

```
        RUN-LCM89                          74/11/14    16.32.37    V5CLARK1HA   PAGE NO. 1

              SUBROUTINE FFTBXY(P)
         3    DIMENSION P(4096,2)
         3    CALL FFT2(P(1,1),P(1,2),4096,1)
         7    RETURN
        10    END
        CD FUS


SUBPROGRAM LENGTH - FFTBXY
    20

VARIABLE ASSIGNMENTS
NAME    - LOCATION

P       ,R      0

EXTERNAL ASSIGNMENTS
FFT2    ,R

START OF      -       CONSTANTS       TEMPORARIES       INDIRECTS       -       UNUSED COMPI
                          12              14                16                      746
```

```
     RUN=LCM89                          74/11/14    16.32.37    V5CLARK1HA   PAGE NO. 1

                SUBROUTINE DELSQ(P,C,C64,S,S64,L,N,C6)
  20            DIMENSION P(256,64,2),C(255),S(255),C64(255),S64(255)
      C         DISPLAY OFF
  20            DO 10 M=1,256
  21            CALL FFT2(P(M,1,1),P(M,1,2),64,-256)
  34         10 CONTINUE
  36            DO 20 K=1,64
  37            K2=2*K-1
  40            K3=3*K-2
  42            K4=4*K-3
  45            DO 20 J=1,4
  46            J1=J+L*N-1
  50            J2=2*J1-1
  53            J3=3*J1-2
  55            J4=4*J1-3
  57            DO 20 I=1,64
  61            I2=2*I-1
  62            I3=3*I-2
  64            I4=4*I-3
  66            X=C6*(2.*(C(I4)*C64(J4)-S(I4)*S64(J4)+C64(I4)+C64(K4))+128.*(C(I2)
               2*C64(J2)-S(I2)*S64(J2)+C64(I2)+C64(K2))-32.*(C(I3)*C64(J3)-S(I3)*
               3S64(J3)+C64(I3)+C64(K3))+32.*(C(I)*C64(J1)-S(I)*S64(J1)+C64(I)+
               4C64(K))-390.)
 151            M=I+(J-1)*64
 155            IF(ABS(X).LT.1.E-05)GO TO 17
 161            X=1./X
 162            GO TO 18
 163         17 X=0.
 171         18 P(M,K,1)=X*P(M,K,1)
      C         DISPLAY ON IF((K.EQ.1).AND.(I.EQ.1))
 172            P(M,K,2)=X*P(M,K,2)
      C         DISPLAY OFF
      C         IF((K.EQ.1).AND.(I.EQ.1))PRINT 51,K,I,P(M,K,1)
      C      51 FORMAT(*  P(M,K,1)=*E14.7)
 176         20 CONTINUE
 204            DO 30 M=1,256
 206            CALL FFT2(P(M,1,1),P(M,1,2),64,256)
 221         30 CONTINUE
 223            RETURN
 224            END


SUBPROGRAM LENGTH - DELSQ
   300

STATEMENT ASSIGNMENTS
STMT NO.  LOCATION          STMT NO.  LOCATION
17        164               18        165

VARIABLE ASSIGNMENTS
NAME   .  LOCATION          NAME   .  LOCATION          NAME   .  LOCATION          NAME

C      .R      1            C6     .R      7            C64    .R      2            I
I2     .I    262            I3     .I    263            I4     .I    264            J
```

```
      RUN-LCM89                          74/11/14     16.32.37     V5CLARK1HA    PAGE NO. 1

                SUBROUTINE DVDT(J,V,W,DU,DV,DW,P,M,CN,C8,C11,RMSU,RMSV,RMSW)
     26         DIMENSION U(6400),V(6400),W(6400),DU(1024),DV(1024),DW(1024),
                2P(6400)
     26         DO 10 I=1,1024
     56         DU(I)=U(I+2688)-C8*(P(I+2686)-P(I+2690)+8.*(P(I+2689)-P(I+2687)))
     66         DV(I)=V(I+2688)-C8*(P(I+2560)-P(I+2816)+8.*(P(I+2752)-P(I+2624)))
     77      10 DW(I)=W(I+2688)-C8*(P(I+128)-P(I+5248)+8.*(P(I+3968)-P(I+1408)))
    141         RETURN
    142         END


SUBPROGRAM LENGTH - DVDT
   203

VARIABLE ASSIGNMENTS
NAME    *  LOCATION          NAME    *  LOCATION          NAME    *  LOCATION          NAME    *

CN      *R      10          C11     *R      12          C8      *R      11          DU      *
DV      *R       4          DW      *R       5          I       *I     202          M       *
P       *R       6          RMSU    *R      13          RMSV    *R      14          RMSW
U       *R       0          V       *R       1          W       *R       2

START OF     -       CONSTANTS          TEMPORARIES          INDIRECTS     -       UNUSED COMPIL
                        144                  145                  163                   743
```

```
      SUBROUTINE CALCPR(U,V,W,P,M,C3,C7)
17    DIMENSION U(6400),V(6400),W(6400),P(4096)
C     DISPLAY OFF
17    IF(M.EQ.2)GO TO 20
20    I1=1
21    I2=1024
22    IF(M.EQ.1)J=2944
25    IF(M.EQ.3)J=896
30    IF(M.EQ.4)J=1920
33   5 DO 10 I=I1,I2
37    JJ=I+J
40    P(JJ)=C7*P(JJ)+C3*(U(I+2686)-U(I+2690)+V(I+2560)-V(I+2816)+W(I+128
     2)-W(I+5248)+8.*(J(I+4609)-U(I+5687)+V(I+4752)-V(I+2624)+W(I+3968)
     3-W(I+4408)))
65   10 CONTINUE
67    IF(M.NE.2)GO TO 50
74    GO TO 3
74   20 I1=1
75    I2=128
76    J=3968
77    GO TO 5
100  30 IF(J.EQ.-128)GO TO 50
102   I1=129
103   I2=1024
104   J=-128
105   GO TO 5
106  50 RETURN
107   END
```

SUBPROGRAM LENGTH - CALCPR
142

STATEMENT ASSIGNMENTS

STMT NO→	LOCATION	STMT NO→	LOCATION	STMT NO→	LOCATION	STMT NO→
5 →	34	20 →	75	30 →	101	50 →

VARIABLE ASSIGNMENTS

NAME →	LOCATION	NAME →	LOCATION	NAME →	LOCATION	NAME →
C3 →R	5	C7 →R	6	I →I	135	I1 →
I2 →I	137	J →I	140	JJ →I	141	M →
P →R	3	U →R	0	V →R	1	W →

START OF -	CONSTANTS	TEMPORARIES	INDIRECTS -	UNUSED COMPI
	111	112	120	743

```
      SUBROUTINE DVDTMP(U,V,W,DU,DV,DW,M,CN,C8,C11,C12,ITIME,RMSD,RMSU,
     2RMSV,RMSW,SK1,SK2,UMAX,VMAX,WMAX,S,ISET)
37    DIMENSION U(64,20,5),V(6400),W(6400),DU(1024),DV(1024),DW(1024),
     2P(6400),S(64)
37    C11=.5*C8
41    C2=C11*CN
43    IF(ISET.EQ.1)GO TO 20
46    DO 10 I=1,1024
67    DIV=U(I+2686)-U(I+2690)+V(I+2560)-V(I+2816)+W(I+128)-W(I+5248)
     2+8.*(U(I+2689)-U(I+2687)+V(I+2752)-V(I+2624)+W(I+3968)-W(I+1408))
107   DU(I)=-C1*(U(I+2688)*(U(I+2686)-U(I+2690)+8.*(U(I+2689)-U(I+2687))
     2)+V(I+2688)*(U(I+2560)-U(I+2816)+8.*(U(I+2752)-U(I+2624)))
     3+W(I+2688)*(U(I+128)-U(I+5248)+8.*(U(I+3968)-U(I+1408)))
     4+U(I+2686)**2-U(I+2690)**2+8.*(U(I+2689)**2-U(I+2687)**2)
     5+U(I+2560)*V(I+2560)-U(I+2816)*V(I+2816)+8.*(U(I+2752)*V(I+2752)
     6-U(I+2624)*V(I+2624))+U(I+128)*W(I+128)-U(I+5248)*W(I+5248)+8.*
     7(U(I+3968)*W(I+3968)-U(I+1408)*W(I+1408))+U(2688)*DIV)
     9+C2*(16.*(U(I+2689)+U(I+2687)+U(I+2752)+U(I+2624)+U(I+3968)+U(I+
     A1408))-U(I+2686)-U(I+2690)-U(I+2816)-U(I+2560)-U(I+128)-U(I+5248)
     B-90.*U(I+2688))
224   DV(I)=-C1*(U(I+2688)*(V(I+2686)-V(I+2690)+8.*(V(I+2689)-V(I+2687))
     2)+V(I+2688)*(V(I+2560)-V(I+2816)+8.*(V(I+2752)-V(I+2624)))
     3+W(I+2688)*(V(I+128)-V(I+5248)+8.*(V(I+3968)-V(I+1408)))
     4+V(I+2686)*U(I+2686)-V(I+2690)*U(I+2690)+8.*(V(I+2689)*U(I+2689)
     5-V(I+2687)*U(I+2687))+V(I+2560)**2-V(I+2816)**2+8.*(V(I+2752)**2
     6-V(I+2624)**2)+V(I+128)*W(I+128)-V(I+5248)*W(I+5248)+8.*(V(I+3968)
     7*W(I+3968)-V(I+1408)*W(I+1408))+V(2688)*DIV)
     9+C2*(16.*(V(I+2689)+V(I+2687)+V(I+2752)+V(I+2624)+V(I+3968)+V(I+
     A1408))-V(I+2686)-V(I+2690)-V(I+2816)-V(I+2560)-V(I+128)-V(I+5248)
     B-90.*V(I+2688))
356   DW(I)=-C1*(U(I+2688)*(W(I+2686)-W(I+2690)+8.*(W(I+2689)-W(I+2687))
     2)+V(I+2688)*(W(I+2560)-W(I+2816)+8.*(W(I+2752)-W(I+2624)))
     3+W(I+2688)*(W(I+128)-W(I+5248)+8.*(W(I+3968)-W(I+1408)))
     4+ W(I+2686)*U(I+2686)-W(I+2690)*U(I+2690)+8.*(W(I+2689)*U(I+2689)
     5-W(I+2687)*U(I+2687))+W(I+2560)*V(I+2560)-W(I+2816)*V(I+2816)+8.*
     6(W(I+2752)*V(I+2752)-W(I+2624)*V(I+2624))+W(I+128)**2-W(I+5248)**2
     7+8.*(W(I+3968)**2-W(I+1408)**2)+W(2688)*DIV)
     9+C2*(16.*(W(I+2689)+W(I+2687)+W(I+2752)+W(I+2624)+W(I+3968)+W(I+
     A1408))-W(I+2686)-W(I+2690)-W(I+2816)-W(I+2560)-W(I+128)-W(I+5248)
     B-90.*W(I+2688))
537   10 CONTINUE
541      GO TO 50
541   20 DO 25 I=1,1024
563      DIV=U(I+2686)-U(I+2690)+V(I+2560)-V(I+2816)+W(I+128)-W(I+5248)
        2+8.*(U(I+2689)-U(I+2687)+V(I+2752)-V(I+2624)+W(I+3968)-W(I+1408))
603      DU(I)=-C1*(U(I+2688)*(U(I+2686)-U(I+2690)+8.*(U(I+2689)-U(I+2687))
        2)+V(I+2688)*(U(I+2560)-U(I+2816)+8.*(U(I+2752)-U(I+2624)))
        3+W(I+2688)*(U(I+128)-U(I+5248)+8.*(U(I+3968)-U(I+1408)))
        4+U(I+2686)**2-U(I+2690)**2+8.*(U(I+2689)**2-U(I+2687)**2)
        5+U(I+2560)*V(I+2560)-U(I+2816)*V(I+2816)+8.*(U(I+2752)*V(I+2752)
        6-U(I+2624)*V(I+2624))+U(I+128)*W(I+128)-U(I+5248)*W(I+5248)+8.*
        7(U(I+3968)*W(I+3968)-U(I+1408)*W(I+1408))+U(2688)*DIV)
        9+C2*(16.*(U(I+2689)+U(I+2687)+U(I+2752)+U(I+2624)+U(I+3968)+U(I+
        A1408))-U(I+2686)-U(I+2690)-U(I+2816)-U(I+2560)-U(I+128)-U(I+5248)
        B-90.*U(I+2688))
```

```
  720       DV(I)=-C1*(U(I+2688)*(V(I+2686)-V(I+2690)+8.*(V(I+2689)-V(I+2687))
           2)+V(I+2688)*(V(I+2560)-V(I+2816)+8.*(V(I+2752)-V(I+2624)))
           3+W(I+2688)*(V(I+128)-V(I+5248)*8.*(V(I+3968)-V(I+1408)))
           4+V(I+2686)*U(I+2686)-V(I+2690)*U(I+2690)+8.*(V(I+2689)*U(I+2689)
           5-V(I+2687)*U(I+2687))+V(I+2560)**2-V(I+2816)**2+8.*(V(I+2752)**2
           6-V(I+2624)**2)+V(I+128)*W(I+128)-V(I+5248)*W(I+5248)+8.*(V(I+3968)
           7*W(I+3968)-V(I+1408)*W(I+1408))*V(2688)*DIV)
           9+C2*(16.*(V(I+2689)+V(I+2687)+V(I+2752)+V(I+2624)+V(I+3968)+V(I+
           A1408))-V(I+2686)-V(I+2690)-V(I+2816)-V(I+2560)-V(I+128)-V(I+5248)
           B-90.*V(I+2688))
 1057       DW(I)=-C1*(U(I+2688)*(W(I+2686)-W(I+2690)+8.*(W(I+2689)-W(I+2687))
           2)+V(I+2688)*(W(I+2560)-W(I+2816)+8.*(W(I+2752)-W(I+2624)))
           3+W(I+2688)*(W(I+128)-W(I+5248)+8.*(W(I+3968)-W(I+1408)))
           4+ W(I+2686)*U(I+2686)-W(I+2690)*U(I+2690)+8.*(W(I+2689)*U(I+2689)
           5-W(I+2687)*U(I+2687))+W(I+2560)*V(I+2560)-W(I+2816)*V(I+2816)+8.*
           6(W(I+2752)*V(I+2752)-W(I+2624)*V(I+2624))+W(I+128)**2-W(I+5248)**2
           7+8.*(W(I+3968)**2-W(I+1408)**2)+W(2688)*DIV)
           9+C2*(16.*(W(I+2689)+W(I+2687)+W(I+2752)+W(I+2624)+W(I+3968)+W(I+
           A1408))-W(I+2686)-W(I+2690)-W(I+2816)-W(I+2560)-W(I+128)-W(I+5248)
           B-90.*W(I+2688))
 1177       RMSD=RMSD+DIV**2
 1200       X=U(I+2688)**2
 1202       RMSU=RMSU+X
 1204       IF(X.GT.UMAX)UMAX=X
 1213       X=V(I+2688)**2
 1215       RMSV=RMSV+X
 1217       IF(X.GT.VMAX)VMAX=X
 1223       X=W(I+2688)**2
 1225       RMSW=RMSW+X
 1227       IF(X.GT.WMAX)WMAX=X
 1234       X=C12*(U(I+2686)-U(I+2690)+8.*(U(I+2689)-U(I+2687)))
 1243       SK1=SK1+X**3
 1246       SK2=SK2+X**2
 1253    25 CONTINUE
 1255    27 IE=IE+1
 1257       IF(MOD(IE,16).NE.1)GO TO 50
 1263       DO 30 J=3,18
 1264       DO 30 I=1,64
 1265       DO 30 L=1,64
 1266       LL=L+I-1
 1270       IF(LL.GT.64)LL=LL-64
 1273    30 S(I)=S(I)+U(L,J,3)*U(LL,J,3)
 1312    50 RETURN
 1313       END
```

SUBPROGRAM LENGTH - DVDTMP
16172

STATEMENT ASSIGNMENTS

STMT NO.	LOCATION	STMT NO.	LOCATION	STMT NO.	LOCATION	STMT NO
20	542	27	1256	30	1274	50

VARIABLE ASSIGNMENTS

START

The program START creates the initial velocity field with the correct total energy and energy spectrum and a small (but not zero) divergence.

```
      PROGRAM START(OUT,FSET7)
      IMPLICIT INTEGER (Z)
      LCM/BB1/UR
      LCM/BB2/UI
      DIMENSION VR(256,64),VI(256,64)
      DIMENSION Q(64,67,8),RQ1(20),RQ2(20),
     2VRS(64,64,3),VIS(64,64,3)        ,DLG(64,64,16),UR(64,16,64),UI(64,
     316,64),E(55),XK(55)
      DIMENSION ITAPE(5),ITAPE(2)
      EQUIVALENCE (VRS(1),UR(1)),(VIS(1),UI(1)),(DLG(1),UR(1))
      EQUIVALENCE (Q(1),VR(1)),(Q(16385),VI(1))
      DATA XK/.31,.44,.54,.626,.70,.766,.83,.88,.941,.992,1.086,1.175,
     21.256,1.35,1.45,1.55,1.65,1.75,1.85,2.0,2.2,2.4,2.6,2.8,3.0,3.2,
     33.4,3.6,3.8,4.0,4.2,4.4,4.6,4.8,5.0,5.5,6.,6.5,7.,7.5,8.,8.5,9.,
     410.,11.,12.,13.,14.,15.,16.,17.,18.,19.,20.,30./
      DATA E/0.,52.7,121.6,101.4,87.5,79.1,67.9,59.4,51.5,45.1,39.8,
     229.1,22.6,17.7,13.2,10.,7.91,6.32,5.06,4.09,3.,2.07,1.39,1.09,
     3.88,.70,.58,.48,.39,.32,.26,.22,.18,.14,.11,.088,.062,.039,.025,
     4.016,.011,.008,.0057,.0042,.0024,.0014,.00087,.00053,.00034,
     5.00022,.00014,.000084,.000047,.000023,.00001/
      DATA W1,W2,D1/2L-1,2LW2,2LD1/
      DATA RQ1/20*0./,RQ2/20*0./
      DATA ITAPE/1,4LTAPE/,I7TAPE/4,3*0,8LXX0O3593/
      CALL MEMREQ(13107,1)
      CALL FORQTS(W1,RQ1)
      CALL FORQTS(W2,RQ2)
      CALL CREATF(W1,U,RT,0,0,0,0,0,0,1536)
      CALL CREATE(W2,U,RT,0,0,0,0,0,0,1536)
      C=SQRT(3.1415927/(20.*4096.))*3.1415927*3.5859
      CC=6.28318/20.
      RMS=0.
      IX=0
      X=0.
      CALL OPEN(5,FSET7,0,234000B,0,0,0,3200,ITAPE)
      DO 50 Z=1,64
      X3=Z-1
      IF(X3.GT.31.1)X3=X3-64.
      DO 20 J=1,64
      X2=J-1
      IF(X2.GT.31.1)X2=X2-64.
      DO 20 I=1,64
      X1=I-1
      IF(X1.GT.31.1)X1=X1-64.
      B=SQRT(X1**2+X2**2)
      IF(B.LT..001)GO TO 12
      P2=X1/B
      P1=-X2/B
      B=SQRT((X3*P1)**2+(X3*P2)**2+(X2*P1)**2-2.*X1*X2*P1*P2+(X1*P2)**2)
      Q1=-P2*X3/B
      Q2=P1*X3/B
      Q3=(P2*X1-P1*X2)/B
      GO TO 13
   12 P1=1.
      P2=0.
      Q1=0.
      Q2=1.
      Q3=0.
   13 Y=CC*SQRT(X1**2+X2**2+X3**2)
      DO 14 N=1,55
      IF(Y.GT.XK(N))GO TO 14
```

```
      M=N
      GO TO 15
   14 CONTINUE
      A=0.
      GO TO 16
   15 A=SQRT(E(M))
   16 CONTINUE
      THETA1=RANF(0.)*6.2831854
      THETA2=RANF(0.)*6.2831854
      C1=COS(THETA1)
      C2=COS(THETA2)
      S1=SIN(THETA1)
      S2=SIN(THETA2)
      Q (I,J,1)=A*(C1*P1+S1*Q1)
      Q (I,J,2)=A*(C1*P2+S1*Q2)
      Q (I,J,3)=A*S1*Q3
      Q (I,J,4)=A*(C2*P1+S2*Q1)
      Q (I,J,5)=A*(C2*P2+S2*Q2)
      Q (I,J,6)=A*S2*Q3
   20 CONTINUE
      DO 30 K=1,3
      KK=K+3
      DO 30 J=1,64
      CALL FFT2(Q(1,J,K),Q(1,J,KK),64,1)
   30 CONTINUE
      DO 33 K=1,3
      KK=K+3
      DO 33 I=1,64
      CALL FFT2(Q(I,1,K),Q(I,1,KK),64,64)
   33 CONTINUE
      NSECT=(Z-1)*24
   31 I=IDONE(W1)
      IF(I.NE.1)GO TO 31
   32 I=IDONE(W2)
      IF(I.NE.1)GO TO 32
      SMALL OUT(Q(1,1,1),DLG(1,1,4),12288)
      SMALL OUT(Q(1,1,4),DLG(1,1,7),12288)
      I=IRANW(RQ1,DLG(1,1,4),12288,NSECT,W1)
      I=IRANW(RQ2,DLG(1,1,7),12288,NSECT,W2)
   50 CONTINUE
   47 I=IDONE(W1)
      IF(I.NE.1)GO TO 47
   48 I=IDONE(W2)
      IF(I.NE.1)GO TO 48
C     READ 64 I S BY 16 J S BY 64 Z S INTO LARGE CORE
      NINC=0
      DO 200 N=1,4
      NSECT1=NINC
      DO 150 M=1,3
      NSECT=NSECT1
      DO 80 Z=1,64
      I=IRANR(RQ1,UR(1,1,Z),1024,NSECT,W1)
      I=IRANR(RQ2,UI(1,1,Z),1024,NSECT,W2)
   81 I=IDONE(W1)
      IF(I.NE.1)GO TO 81
   82 I=IDONE(W2)
      IF(I.NE.1)GO TO 82
   80 NSECT=NSECT+24
      DO 95 J=1,16,4
      DO 90 Z=1,64
```

```
      SMALL IN (VR(1,Z),UR(1,J,Z),256)
   90 SMALL IN (VI(1,Z),UI(1,J,Z),256)
      DO 91 I=1,256
   91 CALL FFT2(VR(I,1),VI(I,1),64,256)
      DO 92 Z=1,64
      SMALL OUT(VR(1,Z),UR(1,J,Z),256)
   92 SMALL OUT(VI(1,Z),UI(1,J,Z),256)
   95 CONTINUE
      NSECT=NSECT1
      DO 109 Z=1,64
      I=IRANW(RQ1,UR(1,1,Z),1024,NSECT,W1)
  101 I=IDONE(W1)
      IF(I.NE.1)GO TO 101
  100 NSECT=NSECT+2
  150 NSECT1=NSECT1+8
  200 NINC=NINC+2
      DO 320 K=1,12
      NSECT=(K-1)*128
      I=IRANR(RQ1,UR(1),65536,NSECT,W1)
  321 I=IDONE(W1)
      IF(I.NE.1)GO TO 321
      DO 310 M=1,2
      II=32768/64*(M-1)+1
      SMALL IN(Q(1),UR(II),32768)
      DO 300 I=1,32768
      Q(I)=C*Q(I)
  300 RMS=RMS+Q(I)**2
  310 WRITE(7),(Q(I),I=1,32768)
  320 CONTINUE
      RMS=SQRT(RMS/3.)/512.
      PRINT 350,RMS
  350 FORMAT(' RMS=',E14.7)
      CALL AFSREL(SLFSET7,0,I7TAPE)
      STOP
      END
```